Practitioner' Docket N . 915-006.029 ***PATENT***

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: E. KAAPPA **Group No.:** To Be Assigned
Application No.: 0 To / Be Assigned
Filed: Herewith **Examiner:** To Be Assigned
For: Method and Device for Defining Objects Allowing Establishment of a Device Management Tree for Mobile Communication Devices

Assistant Commissioner for Patents
Washington, D.C. 20231

TRANSMITTAL OF CERTIFIED COPY

Attached please find the certified copy of the foreign application from which priority is claimed for this case:

Country:	International/WIPO
Application Number:	PCT/IB02/04857
Filing Date:	November 21, 2002

WARNING: "When a document that is required by statute to be certified must be filed, a copy, including a photocopy or facsimile transmission of the certification is not acceptable." 37 C.F.R. § 1.4(f) (emphasis added).

SIGNATURE OF PRACTITIONER

Reg. No. 27,550

Alfred A. Fressola
(type or print name of practitioner)

Tel. No. (203) 261-1234

Ware, Fressola, Van Der Sluys & Adolphson LLP

P.O. Address Bradford Green, Building Five
755 Main Street, P.O. Box 224
Monroe, CT 06468

Customer No.: 004955

NOTE: The claim to priority need be in no special form and may be made by the attorney or agent, if the foreign application is referred to in the oath or declaration, as required by § 1.63.

CERTIFICATE OF MAILING (37 C.F.R. § 1.8a)

I hereby certify that this correspondence is, on the date shown below is being deposited with the United States Postal Service with sufficient postage as ~~first class~~ Express mail in an envelope addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

Express Mail No. EV 303712723 US

Signature

Date: November 21, 2003

Annemarie Maher
(type or print name of person certifying)

(Transmittal of Certified Copy [5-4])

WORLD INTELLECTUAL PROPERTY ORGANIZATION
ORGANISATION MONDIALE DE LA PROPRIÉTÉ INTELLECTUELLE
34, chemin des Colombettes, Case postale 18, CH-1211 Genève 20 (Suisse)
Téléphone: (41 22) 338 91 11 - e-mail: wipo.mail @ wipo.int. - Fac-similé: (41 22) 733 54 28

PATENT COOPERATION TREATY (PCT)
TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

CERTIFIED COPY OF THE INTERNATIONAL APPLICATION AS FILED AND OF ANY CORRECTIONS THERETO

COPIE CERTIFIÉE CONFORME DE LA DEMANDE INTERNATIONALE, TELLE QU'ELLE A ÉTÉ DÉPOSÉE, AINSI QUE DE TOUTES CORRECTIONS Y RELATIVES

International Application No.
Demande internationale n° **PCT/IB02/04857**

International Filing Date
Date du dépôt international **21 November 2002 (21.11.02)**

Geneva/Genève,

31 October 2003 (31.10.03)

International Bureau of the World Intellectual Property Organization (WIPO)

Bureau International de l'Organisation Mondiale de la Propriété Intellectuelle (OMPI)

BUREAU INTERNATIONAL DE L'ORGANISATION MONDIALE DE LA PROPRIETE INTELLECTUELLE
INTERNATIONAL BUREAU OF THE WORLD INTELLECTUAL PROPERTY ORGANIZATION

J.-L. Baron
Head, PCT Receiving Office Section
Chef de la section "office récepteur du PCT"

RECORD COPY

PCT

REQUEST

The undersigned requests that the present international application be processed according to the Patent Cooperation Treaty.

For receiving Office use only

International Application No. PCT / IB 0 2 / 0 4 8 5 7

International Filing Date 21 NOVEMBER 2002 (2 1. 1 1. 0 2)

INTERNATIONAL BUREAU OF WIPO
PCT International Application
Name of receiving Office and "PCT International Application"

Applicant's or agent's file reference *(if desired) (12 characters maximum)* 51300 WO

Box No. I TITLE OF INVENTION Method and device for defining objects allowing to establish a device management tree for mobile communication devices

Box No. II APPLICANT ☐ This person is also inventor

Name and address: *(Family name followed by given name; for a legal entity, full official designation. The address must include postal code and name of country. The country of the address indicated in this Box is the applicant's State (that is, country) of residence if no State of residence is indicated below.)*

Nokia Corporation
Keilalahdentie 4
02150 Espoo
Finland

Telephone No.

Facsimile No.

Teleprinter No.

Applicant's registration No. with the Office

State *(that is, country)* of nationality: Finland

State *(that is, country)* of residence: Finland

This person is applicant for the purposes of: ☐ all designated States ☒ all designated States except the United States of America ☐ the United States of America only ☐ the States indicated in the Supplemental Box

Box No. III FURTHER APPLICANT(S) AND/OR (FURTHER) INVENTOR(S)

Name and address: *(Family name followed by given name; for a legal entity, full official designation. The address must include postal code and name of country. The country of the address indicated in this Box is the applicant's State (that is, country) of residence if no State of residence is indicated below.)*

Kaappa Eero
Pyynikintie 17 c 31
33230 Tampere
Finland

This person is:
☐ applicant only
☒ applicant and inventor
☐ inventor only *(If this check-box is marked, do not fill in below.)*

Applicant's registration No. with the Office

State *(that is, country)* of nationality: Finland

State *(that is, country)* of residence: Finland

This person is applicant for the purposes of: ☐ all designated States ☐ all designated States except the United States of America ☒ the United States of America only ☐ the States indicated in the Supplemental Box

☐ Further applicants and/or (further) inventors are indicated on a continuation sheet.

Box No. IV AGENT OR COMMON REPRESENTATIVE; OR ADDRESS FOR CORRESPONDENCE

The person identified below is hereby/has been appointed to act on behalf of the applicant(s) before the competent International Authorities as: ☒ agent ☐ common representative

Name and address: *(Family name followed by given name; for a legal entity, full official designation. The address must include postal code and name of country.)*

Kurig, Thomas (Dr.)
Becker, Kurig, Straus
Bavariastrasse 7
D-80336 München

Telephone No. 089-746 303 0

Facsimile No. 089-746 303 11

Teleprinter No.

Agent's registration No. with the Office

☐ **Address for correspondence:** Mark this check-box where no agent or common representative is/has been appointed and the space above is used instead to indicate a special address to which correspondence should be sent.

Box No. V DESIGNATION OF STATES *Mark the applicable check-boxes below; at least one must be marked.*

The following designations are hereby made under Rule 4.9(a):

Regional Patent

☒ **AP** **ARIPO Patent:** GH Ghana, GM Gambia, KE Kenya, LS Lesotho, MW Malawi, MZ Mozambique, SD Sudan, SL Sierra Leone, SZ Swaziland, TZ United Republic of Tanzania, UG Uganda, ZM Zambia, ZW Zimbabwe, and any other State which is a Contracting State of the Harare Protocol and of the PCT *(if other kind of protection or treatment desired, specify on dotted line)* .....

☒ **EA** **Eurasian Patent:** AM Armenia, AZ Azerbaijan, BY Belarus, KG Kyrgyzstan, KZ Kazakhstan, MD Republic of Moldova, RU Russian Federation, TJ Tajikistan, TM Turkmenistan, and any other State which is a Contracting State of the Eurasian Patent Convention and of the PCT

☒ **EP** **European Patent:** AT Austria, BE Belgium, BG Bulgaria, CH & LI Switzerland and Liechtenstein, CY Cyprus, CZ Czech Republic, DE Germany, DK Denmark, EE Estonia, ES Spain, FI Finland, FR France, GB United Kingdom, GR Greece, IE Ireland, IT Italy, LU Luxembourg, MC Monaco, NL Netherlands, PT Portugal, SE Sweden, SK Slovakia, TR Turkey, and any other State which is a Contracting State of the European Patent Convention and of the PCT

☒ **OA** **OAPI Patent:** BF Burkina Faso, BJ Benin, CF Central African Republic, CG Congo, CI Côte d'Ivoire, CM Cameroon, GA Gabon, GN Guinea, GQ Equatorial Guinea, GW Guinea-Bissau, ML Mali, MR Mauritania, NE Niger, SN Senegal, TD Chad, TG Togo, and any other State which is a member State of OAPI and a Contracting State of the PCT *(if other kind of protection or treatment desired, specify on dotted line)* .....

National Patent *(if other kind of protection or treatment desired, specify on dotted line)*:

☒ AE United Arab Emirates	☒ GM Gambia	☒ NZ New Zealand
☒ AG Antigua and Barbuda	☒ HR Croatia	☒ OM Oman
☒ AL Albania	☒ HU Hungary	☒ PH Philippines
☒ AM Armenia	☒ ID Indonesia	☒ PL Poland
☒ AT Austria	☒ IL Israel	☒ PT Portugal
☒ AU Australia	☒ IN India	☒ RO Romania
☒ AZ Azerbaijan	☒ IS Iceland	☒ RU Russian Federation
☒ BA Bosnia and Herzegovina	☒ JP Japan	
☒ BB Barbados	☒ KE Kenya	☒ SD Sudan
☒ BG Bulgaria	☒ KG Kyrgyzstan	☒ SE Sweden
☒ BR Brazil	☒ KP Democratic People's Republic of Korea	☒ SG Singapore
☒ BY Belarus		☒ SI Slovenia
☒ BZ Belize	☒ KR Republic of Korea	☒ SK Slovakia
☒ CA Canada	☒ KZ Kazakhstan	☒ SL Sierra Leone
☒ CH & LI Switzerland and Liechtenstein	☒ LC Saint Lucia	☒ TJ Tajikistan
☒ CN China	☒ LK Sri Lanka	☒ TM Turkmenistan
☒ CO Colombia	☒ LR Liberia	☒ TN Tunisia
☒ CR Costa Rica	☒ LS Lesotho	☒ TR Turkey
☒ CU Cuba	☒ LT Lithuania	☒ TT Trinidad and Tobago
☒ CZ Czech Republic	☒ LU Luxembourg	
☒ DE Germany	☒ LV Latvia	☒ TZ United Republic of Tanzania
☒ DK Denmark	☒ MA Morocco	☒ UA Ukraine
☒ DM Dominica	☒ MD Republic of Moldova	☒ UG Uganda
☒ DZ Algeria		☒ US United States of America
☒ EC Ecuador	☒ MG Madagascar	
☒ EE Estonia	☒ MK The former Yugoslav Republic of Macedonia	☒ UZ Uzbekistan
☒ ES Spain		☒ VN Viet Nam
☒ FI Finland	☒ MN Mongolia	☒ YU Yugoslavia
☒ GB United Kingdom	☒ MW Malawi	☒ ZA South Africa
☒ GD Grenada	☒ MX Mexico	☒ ZM Zambia
☒ GE Georgia	☒ MZ Mozambique	☒ ZW Zimbabwe
☒ GH Ghana	☒ NO Norway	

Check-boxes below reserved for designating States which have become party to the PCT after issuance of this sheet:

☒ VC Saint Vincent and the Grenadines	☐	☐
☒ SC Seychelles	☐	☐

Precautionary Designation Statement: In addition to the designations made above, the applicant also makes under Rule 4.9(b) all other designations which would be permitted under the PCT except any designation(s) indicated in the Supplemental Box as being excluded from the scope of this statement. The applicant declares that those additional designations are subject to confirmation and that any designation which is not confirmed before the expiration of 15 months from the priority date is to be regarded as withdrawn by the applicant at the expiration of that time limit. *(Confirmation (including fees) must reach the receiving Office within the 15-month time limit.)*

Box No. VI PRIORITY CLAIM

The priority of the following earlier application(s) is hereby claimed:

Filing date of earlier application *(day/month/year)*	Number of earlier application	Where earlier application is:		
		national application: country or Member of WTO	regional application:* regional Office	international application: receiving Office
item (1)				
item (2)				
item (3)				
item (4)				
item (5)				

☐ Further priority claims are indicated in the Supplemental Box.

The receiving Office is requested to prepare and transmit to the International Bureau a certified copy of the earlier application(s) *(only if the earlier application was filed with the Office which for the purposes of this international application is the receiving Office)* identified above as:

☐ all items ☐ item (1) ☐ item (2) ☐ item (3) ☐ item (4) ☐ item (5) ☐ other, see Supplemental Box

** Where the earlier application is an ARIPO application, indicate at least one country party to the Paris Convention for the Protection of Industrial Property or one Member of the World Trade Organization for which that earlier application was filed (Rule 4.10(b)(ii)):* ....

Box No. VII INTERNATIONAL SEARCHING AUTHORITY

Choice of International Searching Authority (ISA) *(if two or more International Searching Authorities are competent to carry out the international search, indicate the Authority chosen; the two-letter code may be used)*:

ISA / SE

Request to use results of earlier search; reference to that search *(if an earlier search has been carried out by or requested from the International Searching Authority)*:

Date *(day/month/year)* Number Country *(or regional Office)*

Box No. VIII DECLARATIONS

The following **declarations** are contained in Boxes Nos. VIII (i) to (v) *(mark the applicable check-boxes below and indicate in the right column the number of each type of declaration)*: Number of declarations

☐	Box No. VIII (i)	Declaration as to the identity of the inventor	:
☐	Box No. VIII (ii)	Declaration as to the applicant's entitlement, as at the international filing date, to apply for and be granted a patent	:
☐	Box No. VIII (iii)	Declaration as to the applicant's entitlement, as at the international filing date, to claim the priority of the earlier application	:
☐	Box No. VIII (iv)	Declaration of inventorship (only for the purposes of the designation of the United States of America)	:
☐	Box No. VIII (v)	Declaration as to non-prejudicial disclosures or exceptions to lack of novelty	:

Box No. IX CHECK LIST; LANGUAGE OF FILING

This international application contains:

(a) the following number of sheets in paper form:

request (including declaration sheets)	:	4
description (excluding sequence listing part)	:	38
claims	:	5
abstract	:	1
drawings	:	10
Sub-total number of sheets	:	58
sequence listing part of description *(actual number of sheets if filed in paper form, whether or not also filed in computer readable form; see (b) below)*	:	
Total number of sheets	:	58

(b) sequence listing part of description filed in **computer readable form**

(i) ☐ only (under Section 801(a)(i))

(ii) ☐ in addition to being filed in paper form (under Section 801(a)(ii))

Type and number of carriers (diskette, CD-ROM, CD-R or other) on which the sequence listing part is contained *(additional copies to be indicated under item 9(ii), in right column)*: .....

This international application is accompanied by the following item(s) *(mark the applicable check-boxes below and indicate in right column the number of each item)*: — Number of items

1. ☐ fee calculation sheet :
2. ☐ original separate power of attorney :
3. ☐ original general power of attorney :
4. ☐ copy of general power of attorney; reference number, if any: ..... :
5. ☐ statement explaining lack of signature :
6. ☐ priority document(s) identified in Box No. VI as item(s): ..... :
7. ☐ translation of international application into *(language)*: ..... :
8. ☐ separate indications concerning deposited microorganism or other biological material :
9. ☐ sequence listing in computer readable form (indicate also type and number of carriers (diskette, CD-ROM, CD-R or other))
   - (i) ☐ copy submitted for the purposes of international search under Rule 13*ter* only (and not as part of the international application) :
   - (ii) ☐ *(only where check-box (b)(i) or (b)(ii) is marked in left column)* additional copies including, where applicable, the copy for the purposes of international search under Rule 13*ter* :
   - (iii) ☐ together with relevant statement as to the identity of the copy or copies with the sequence listing part mentioned in left column :
10. ☐ other *(specify)*: ..... :

Figure of the drawings which should accompany the abstract: 5a

Language of filing of the international application: English

Box No. X SIGNATURE OF APPLICANT, AGENT OR COMMON REPRESENTATIVE

Next to each signature, indicate the name of the person signing and the capacity in which the person signs (if such capacity is not obvious from reading the request).

Dr. Thomas Kurig (Patent Attorney)

For receiving Office use only

1. Date of actual receipt of the purported international application: 21 NOVEMBER 2002 (21. 11. 02)

2. Drawings: ☐ received: ☐ not received:

3. Corrected date of actual receipt due to later but timely received papers or drawings completing the purported international application: 22 NOVEMBER 2002 (22. 11. 02)

4. Date of timely receipt of the required corrections under PCT Article 11(2):

5. International Searching Authority (if two or more are competent): ISA / SE

6. ☒ Transmittal of search copy delayed until search fee is paid

For International Bureau use only

Date of receipt of the record copy by the International Bureau:

METHOD AND DEVICE FOR DEFINING OBJECTS ALLOWING TO ESTABLISH A DEVICE MANAGEMENT TREE FOR MOBILE COMMUNICATION DEVICES

The present invention relates to a method and device for defining and adding objects in a hierarchical object structure which allows to establish a device management tree structure for storing management related information of a mobile communication device. In particular, the present invention relates to a method for creating one or several objects within the management tree resulting in a management tree which has dynamic structures and which provides an efficient, clear, manageable and preferable organization. Moreover, the present invention relates to devices and systems being adapted to operate the aforementioned method.

The synchronization of data is a well known problem for all users processing same data with at least two different electronic devices. In general, synchronization takes place between a terminal device (e.g., a mobile phone) and a server device (e.g., an application in a local PC or a dedicated synchronization server). Data of portable terminals, such as portable computers, PDA terminals (personal digital assistant), mobile stations or pagers, can be synchronized with network applications, applications of desktop computers or with other databases of the telecommunications system, wherein the term database should be understood as broad as possible, i.e. shall cover arbitrary sets of data. In particular, data of calendar and e-mail applications are typically synchronized.

Synchronization has been based on the use of different manufacturer-specific protocols which are incompatible. This restricts the use of terminal or data types and often causes troubles to the user. In mobile communication, in particular, it is important that data can be retrieved and updated regardless of the terminal and application used.

To improve synchronization of application data, a language known as synchronization markup language SyncML, which is based on the extensible markup language (XML) and a corresponding standardized document type description (DTD), has been developed. By using a SyncML synchronization protocol, which employs messages in the SyncML format, data of any application can be synchronized between networked terminals and a network server of any kind. The SyncML synchronization protocol works both in wireless and in fixed networks and supports several transmission protocols.

The above presented SyncML synchronization technology addresses preferably the synchronization of databases. A problem similar to the synchronization of databases is given by the managing of configuration data and settings necessary for the operation of electronic devices within changing environments, for example of mobile phone operating within mobile communication networks of different network carriers requiring individual carrier related sets of configurations e.g. network access point (NAP) definitions, proxy and gateway server addresses, information and definitions, address information of servers providing certain services such as short message service (SMS), multimedia message service (MMS) and the like. The SyncML device management relates to the harmonizing of such configuration data and settings. The respective configuration data and information is contained in management objects, respectively, associated with respective device feature(s) and operable application(s), respectively.

SyncML device management (SyncML DM) protocol allows management commands to be executed on management objects and it uses a package format similar SyncML synchronization protocol and related definitions and is based also on XML or related XML encoding such as binary encoded XML. A management object might reflect a set of configuration parameters for a device, i.e. configuration parameters of device features and/or configuration parameters and settings of software applications executed on the device. Actions that can be taken against this object might include reading and setting parameter keys and values. Another management object might be the run-time environment for software applications on a device. Actions that can be taken against this type of object might include installing, upgrading, or uninstalling software elements. Preferably, dedicated management servers provide the required configuration parameters, settings, keys and values for synchronization of the device management information aforementioned.

The device management in accordance with the SyncML DM allows to structure the management objects in a hierarchical management tree containing all information which can be managed using the SyncML DM protocol. The management tree is based on a permanent part of the management tree defined and provided by the manufacturer of the respective electronic device supporting SyncML device management. The real management tree present in such an operated electronic device is composed of this permanent part of the management tree which is expanded by one or several dynamically created parts of the management tree. The real management tree is described in some way from a kind of pre-determined tree framework, the device description framework.

It is obviously, that the hierarchical structure of such a management tree containing management information required for operating an electronic device of a type as described above is rather complex depending on the functions provided by the terminal device to a user and the applications operable on the terminal device which all access the management tree for storing, managing and/or retrieving configuration data and settings. Especially, each further function added to the electronic device or application implemented into the electronic device results in an increasing of the total hierarchical structure of the management tree causing a more complicated hierarchical structure. The kind and number of functions and/or applications for improving the operation of the terminal devices is unknown. But it is desired to maintain the introduced hierarchical management tree for storing, retrieving and/or managing configuration data and settings even for future use.

An object of the invention is to provide a method for defining and adding at least one object of a hierarchical object structure constituted by a plurality of individual objects, wherein the hierarchical object structure describes a management tree of an electronic device such that the resulting management tree is structured as efficient and future proofed as possible. An efficient structured management tree provides an efficient management and retrieval of one or more certain management objects for managing, retrieving their contents. Moreover, an efficient structured management tree also takes the limited resources (memory, processing capability etc.) into consideration. The method for defining and adding of one or more objects to be included in the hierarchical object structure, in particular a management tree, is based on a couple of definition rules and regulations the consideration of which results in such an efficient and optimized structured management tree.

The objects of the invention are achieved with a method for adding at least one object into a hierarchical object structure, corresponding systems and device adapted to perform this method, computer programs and software tools which are disclosed in the independent claims. Preferred embodiments of the invention are disclosed in the dependent claims.

According to an embodiment of the invention, a method for adding at least one object into a hierarchical object structure is provided. The hierarchical object structure is constituted by a plurality of objects. The objects are hierarchically associated to each other. The hierarchical association of the objects is obtained by using at least two different types of object, a first object type and a second object type. Both the first object type and a second object type are allowed to have subordinately arranged objects being associated directly therewith.

The at least one object is defined to be associated to a parent object which will be arranged superordinately to and directly associated with the at least one object, i.e. the at least one object to be defined is a child object of the parent object. The parent object is part of the hierarchical object structure. The object type of the parent object is obtained to be compared with the object type of the child object to be defined. In case the object types of both the parent object and the at least one child object agree with each other and both object types match with the first type, an object having the second object type is interposed between the parent object and the child object. The separation ensures, that objects having the first object type are always separated in their hierarchical arrangement by at least one object having the second object type.

According to an embodiment of the invention, the at least two different object types comprises a run-time object type and a fixed object type. The aforementioned first object type corresponds to the run-time object type and the aforementioned second type corresponds to the fixed object type.

According to an embodiment of the invention, at least following object types are suitable for being applied: the fixed object type, run-time object type and leaf object type.

Each object having any one of the aforementioned object types has one directly associated object being arranged superordinately which is denoted as parent object. But objects having fixed object type or run-time object type are allowed to have none, one or more objects being directly associated therewith and being arranged subordinately thereto. The different object types allow to form the hierarchical object structure constituted by a plurality of objects each having one of the described object types.

Accordingly, the aforementioned method for adding at least one object according to an embodiment of the invention may be described alternatively in the following way. The at least one object is defined to be associated to a parent object. The object type of the parent object is obtained to be inspected. In case object type of the parent object is the fixed object type, the at least one object is allowed to have either the fixed object type, the run-time object type or the leaf object type. In case the object type of the parent object is the type run-time object type, the at least one object is allowed to have either the fixed object type or the leaf object type.

The defining of the at least one object may further require a defining of object properties. The object properties may comprise one or more properties out of a group comprising AccessType, DefaultValue, Description, DFFormat, Occurrence, Scope, DFTitle and DFType. The possible object properties is not limited to the presented ones, further different object properties may be also valid.

According to an embodiment of the invention, it is further examined if the parent object has already one or more directly arranged objects being associated subordinately. In case there are existing already one or more objects, the one or more object types of these one or more objects are determined and the obtained one ore more object types are compared with the object type of the child object to be added to the hierarchical object structure. In case it is proved that at least one of the obtained one ore more object types and the object type of the child object have all the first object type (i.e. run-time object type), the child object is added to the parent object under the condition that the child object and the at least one already existing object have a common format.

The format of an object shall define which kind of management related information is allowed and/or suitable for being distributed among the respective object and hierarchically arranged objects being associated subordinately with that respective object. Objects having a common format define that similar management related information being associated with the same device function and/or device application is distributed among these objects and subordinately arranged objects.

According to an embodiment of the invention, the object type of the parent object is determined. Further it is checked whether the parent object has already one or more child objects. In case that the parent object has not any child objects and the parent object and the at least one object to be defined have both the second object types, the parent object and the at least one object are concentrated to a single object having the second object type. The concentrating is performed by replacing the parent object and the at least one object with the single object.

According to an embodiment of the invention, at least a part of a description document is coded in accordance with the at least one object being defined. At least the part of the description document being coded comprises information relating to the at least one object and the properties of the at least one object being defined previously. The coded description document allows to obtain a hierarchical object structure for storing the management related information of an electronic device being distributed among the plurality of objects comprised in the hierarchical structure. The obtaining of one or several objects and at least a part of the hierarchical object structure may be performed by parsing the coded description document or one or more relevant part(s) thereof, respectively.

According to an embodiment of the invention, the hierarchical object structure comprising a plurality of objects describes a hierarchical management structure. The hierarchical management structure may be understood as the hierarchical framework of a database within which in

accordance to the hierarchical structure management related information of an electronic device, especially, of a mobile communication terminal is stored. Certain parts of the management related information being configuration data and/or settings, are associated with or included in at least one of the objects,

According to an embodiment of the invention, the format of an object defines which kind of management related information is allowed and/or suitable for being distributed among the respective object and hierarchically arranged objects being associated subordinately with that respective object. Objects having a common format define that similar management related information being associated with the same device function and/or device application is distributed among these objects and subordinately arranged objects.

According to an embodiment of the invention, the hierarchical object structure comprising the plurality of objects corresponds to a device description framework (DDF) information and/or the hierarchical object structure comprising the plurality of objects corresponds to a management tree employed for device management of an electronic device. The device description framework (DDF) information and/or the management tree are provided and standardized by the SyncML Initiative in form of the synchronization markup language device management (SyncML DM) standard.

According to an embodiment of the invention, the description document corresponds to a device description framework (DDF) document. The DDF document is an extended markup language (XML) encoded document being encoded in accordance with a corresponding description framework document type description (DTD). The extended markup language (XML) is not limited to the below presented clear (plain) text XML encoding but shall cover also binary encoding being based on XML and the like.

According to an embodiment of the invention, a software tool for adding at least one object into a plurality of objects establishing a hierarchical object structure is provided. The software tool comprises program portions for carrying out the operations of the aforementioned methods according to any embodiment of the invention when the software tool is implemented in a computer program and/or executed.

According to an embodiment of the invention, there is provided a computer program for adding at least one object into a plurality of objects establishing a hierarchical object structure. The computer program comprises loadable program code portions for carrying out the operations of

the aforementioned methods according to any embodiment of the invention when the program is executed on a processing device, a computer or a network device.

According to an embodiment of the invention, a computer program product is provided which comprises program code portions stored on a computer readable medium for carrying out the aforementioned methods according to any embodiment of the invention when said program product is executed on a processing device, a computer or network device.

According to an embodiment of the invention, computer data signal is provided. The computer data signal is embodied in a carrier wave and represents a program or program code portions which, when executed by a processor, causes the aforementioned methods according to any embodiment of the invention be carried out.

According to an embodiment of the invention, a processing device comprising a processing unit, a memory unit and a communication interface is provided. The processing unit is coupled to the memory unit and the communication unit allowing for exchanging data in-between them. The processing unit of the processing device is configured allowing for adding at least one object into a hierarchical object structure. The hierarchical object structure is constituted by a plurality of objects. The objects are hierarchically associated to each other. The hierarchical association of the objects is obtained by using at least two different types of object, a first object type and a second object type. Both the first object type and a second object type are allowed to have subordinately arranged objects being associated directly therewith.

The at least one object is defined to be associated to a parent object which will to be arranged superordinately to and directly associated with the at least one object, i.e. the at least one object to be defined is at least one child object of the parent object. The parent object is part of the hierarchical object structure. The object type of the parent object is obtained to be compared with the object type of the at least one object to be defined. In case the object types of both the parent object and the at least one child object agree with each other and both object types match with the first type, an object having the second object type is interposed between the parent object and the at least one object. The separation ensures, that objects having the first object type are always separated in their hierarchical interdependency by at least one object having the second object type.

According to an embodiment of the invention, the processing device is further configured allowing for operating any one of the aforementioned embodiments of methods for adding at least one object into a hierarchical object structure.

According to an embodiment of the invention, a management system is provided which comprises a processing device such as a managed client device or a management server device and a hierarchical object structure. The hierarchical object structure is constituted by a plurality of objects being hierarchically associated. Each object of the plurality of objects has a certain object type which is at least two object types, a first object type and a second object type, are applicable. Each object which has the first object type or the second object type is allowed to have directly arranged objects being associated subordinately therewith.

Each two objects which are arranged hierarchically to each other, which are associated with each other and which have both the first object type are separated by at least one interposed object having the second object type in the hierarchical structure. That is, hierarchically associated objects having the first object type are not allowed to be directly arranged to each other.

The processing device includes at least one management component which is capable to generate at least a part of a hierarchical object structure. The hierarchical arrangement of the hierarchical object structure is mapped thereby to the hierarchical arrangement of the resulting at least part of the hierarchical object structure. The hierarchical object structure serves for retrieving and storing and managing management related information of a managed mobile communication enabled device.

According to an embodiment of the invention, the at least two different object types comprises a run-time parent object type and a fixed parent object type. The aforementioned first object type corresponds to the run-time parent object type and the aforementioned second type corresponds to the fixed parent object type.

According to an embodiment of the invention, at least following object types are suitable for being applied: the fixed parent object type, run-time parent object type and leaf object type.

Each object having any one of the aforementioned object types has one directly associated object being arranged superordinately which is denoted as parent object. But objects having fixed parent object type or run-time parent object type are allowed to have none, one or more objects being directly associated therewith and being arranged subordinately thereto. The different object types allow to form the hierarchical object structure constituted by a plurality of objects each having one of the described object types.

According to an embodiment of the invention, two or more objects which have the first type and which are directly arranged to an object being associated superordinately therewith have a common format. The format of an object shall define which kind of management related information is allowed and/or suitable for being distributed among the respective object and hierarchically arranged objects being associated subordinately with that respective object. Objects having a common format define that similar management related information being associated with the same device function and/or device application is distributed among these objects and subordinately arranged objects.

According to an embodiment of the invention, two directly arranged objects being associated with each other and having both said first type, i.e. a parent object and a child object both having the second type are substituted in the hierarchical object tree with a concentrated object having the second object type. The concentrated object is built of the two replaced objects. That is, the hierarchical object structure comprises a concentrated objects which has been from the above described two directly arranged objects such that an direct arrangement of a parent object and its only child object (both being of the second type) do not occur in the hierarchical object structure.

According to an embodiment of the invention, the processing device is a managed mobile communication enabled device comprising at least a client device management component, i.e. a client management agent, and a storage component. The client device management component allows to generate the at least part of the hierarchical object structure, to establish the part of the hierarchical object structure and to implement the at least part of the hierarchical object structure into the storage component of the managed mobile communication enabled device. The implementing may be an implementing of the generated at least part of the hierarchical object structure into a present existing hierarchical object structure for dynamically extending the hierarchical object structure. The client device management component further is capable to distribute management related information among the plurality of object constituting the hierarchical object structure and to retrieve at least parts of the management related information from one or more object of the plurality of objects for configuring functions of the device and/or applications operable with the device.

According to an embodiment of the invention, the processing device is a management server device comprising at least a server device management component, i.e. a server management agent and/or service management engine, and a communication interface. The server device management component allows to generate the at least part of the hierarchical object structure, to generate one or more management messages including management instructions and corresponding management related data in accordance with the generated at least part of the

hierarchical object structure. The server device management component further is capable to transmit the obtained one or more management messages in conjunction with the communication interface to a managed mobile communication enabled device.

The one or more management messages may instruct the managed mobile communication enabled device to extend its stored hierarchical object structure dynamically in accordance with the transmitted management information, to distribute management related information provided by the server device among the certain objects of the hierarchical object structure, to retrieve management related information from certain objects of the hierarchical object structure being identified (addressed) by the means of address information gained from the generated at least part of the hierarchical object structure and the like.

Invention will be described in greater detail by means of embodiments with reference to the accompanying drawings, in which

Fig. 1a shows a schematic diagram illustrating a set of exemplary electronic devices between which synchronization of information can be operated;

Fig. 1b shows an example overall part of a management tree;

Fig. 2a shows a table illustrating elements used for graphical notation of objects of the device description framework;

Fig. 2b shows a table illustrating special characters used additionally for graphical notation of objects of the device description framework;

Fig. 3a shows a first arrangement of description objects according to an embodiment of the invention;

Fig. 3b shows a second arrangement of description objects according to an embodiment of the invention;

Fig. 3c shows arrangements of description objects to be avoided according to an embodiment of the invention;

Fig. 3d shows an arrangement of description objects to be concentrated to a single description object according to an embodiment of the invention;

Fig. 3e shows an arrangement of description objects to be avoided according to an embodiment of the invention;

Fig. 4a shows a first flow chart illustrating the defining of a description object according to an embodiment of the invention;

Fig. 4b shows a second flow chart illustrating a defining and creating, respectively, of at least a part of a DDF document in accordance with an embodiment of the invention;

Fig. 4c shows a third flow chart illustrating the parsing and generating of at least a part of a management tree being based on a DDF document according to an embodiment of the invention;

Fig. 5a shows an example excerpt of a graphical depiction of a device description framework according to an embodiment of the invention;

Fig. 5b shows a first part of an excerpt of a DDF document corresponding to the graphical depiction shown in Fig. 5a and according to an embodiment of the invention;

Fig. 5c shows a second part of the excerpt of the DDF document shown in Fig. 5b according to an embodiment of the invention;

Fig. 5d shows an example excerpt of a graphical depiction of a management tree in accordance with the graphical depiction of a device description framework illustrated in Fig. 5a according to an embodiment of the invention; and

Fig. 6 shows a block diagram illustrating devices containing components for operating the aforementioned methods according to embodiments of the invention.

In the following, the embodiments of the invention will be described with respect to a system supporting the SyncML device management standard or the related SyncML standard without limiting the invention thereto. Information about the SyncML standard and the SyncML device management (SyncML DM) standard can be obtained from the SyncML Initiative providing publicly the full standard documentation. Same or equal parts, features and/or operations shown in the figures will be referred to using the same reference numerals.

Fig. 1 shows a schematic diagram illustrating a set of exemplary electronic device between which synchronization of information can be operated. A certain database content of preferably mobile terminals shall be harmonized with database content provided by designated devices. Conventionally, mobile terminals act as synchronization clients harmonizing or synchronizing certain pre-defined data with the contents of a database or several databases provided by dedicated server devices. Fig. 1 illustrates a plurality of possible client devices and server devices for the synchronization operation. Typically, client devices are mobile stations like mobile phones 17 or personal digital assistants (PDA), mobile computers like notebooks 15, digital cameras 16 or personal computers (PC). Further, dedicated synchronization server devices may be desktop computers like a personal computer 10, a dedicated network server 11 or even a mobile computer like a notebook 12. It shall be noted that the client device functionality is not limited to mobile terminals as described above although the presented concept of synchronization is described in view of mobile terminals connected to dedicated serving devices.

Whereas the description above referred to a general synchronization and therefore also to the device management synchronization, the following description of the innovative concept will refer explicitly to the SyncML DM protocol.

The SyncML DM service itself is based on the exchange of a management document, which may be divided into a plurality massages or packages, respectively, comprising instructions in order to synchronize the device management data, i.e. the configuration data and settings. SyncML DM protocol comprises two parts: setup phase comprising authentication and device information exchange and management phase. Management phase can be repeated as many times as the server wishes.

Management phase consists of a number of protocol iterations, i.e. protocol iteration means a package from managed client device to management server and a package from management server to managed client device. Content of package sent from the management server to managed client device determines whether the session must be continued or not. If the management server sends management operations in packages that need responses (status or results) from the managed client device, the management phase of the protocol continues with new package from managed client device to management server containing client responses to management operations. Response package from managed client device starts new protocol iteration. The management server can send new management operation package and therefore initiate new protocol iteration as many times as it wishes.

An exemplary and valid total sequence of packages in accordance with the setup phase and the management phase is described in the following section in order to provide a coarse overview of the package exchange.

Package 0 - initiation of the management session: Most managed client devices can receive unsolicited messages, sometimes called "notifications". A management server can use this notification capability to cause the client to initiate a connection back to the management server. several bearers can be used for transmitting management initiation notifications. Note that an identical effect to receiving a management initiation notification can be caused in other ways.

Package 1 - initialization from managed client device to management server: The purpose of the initialization package sent by the managed client device is:

- to send managed client device information (like manufacturer, model etc) to a device management server,
- to identify the managed client device to the management server,

- to inform the server whether the management session was initiated by the server (by sending a trigger in Package 0) or by the client (like end user selecting a menu item).

Package 2 - initialization from management server to managed client device: The purpose of the initialization package sent by the management server is:

- to identify the device management server to the managed client device,
- to send optionally management commands and management data to the managed client device,
- to send optionally further commands like user interaction commands.

Packages 1 and 2 are part of the setup phase of the management process. Following packages 3 and 4 are part of the management phase of the management session.

Package 3 - managed client device response to management server: The purpose of this management package is:

- to transmit results of the management commands sent from the management server to the managed client,
- to transmit results of optional user interaction commands.

Package 4 - further management server operations: The purpose of this management package is:

- to transmit any further necessary management operations or commands from the management server to the managed client, respectively, or
- to close the management session.

A package 4 containing further management operations causes a response of the managed client device in kind of a package 3. Hence, the management session can comprise an arbitrary number of iterations of the packages 3 and 4.

The aforementioned SyncML DM service being based on management message exchanges in accordance with the above described phases and packages, respectively, takes actions against objects constituting the management tree which structures and contains information to be managed, respectively. The management tree is a well-defined hierarchical structured arrangement of all available and used management entities, wherein the position and correspondingly its addressing of certain information contained in a certain management entity reflects the content of the management information, respectively. In view of the above described broad definition of databases, the management tree may be understood as a database for storing configuration data and settings in a well-defined hierarchical organization. The position of

management related information within the management tree is correlated with a certain device function and/or device application employing the respective management related information for operation, respectively. This well-defined hierarchical structure is referred to as management tree. The hierarchical organization of the management objects primarily addresses the need for providing an adequate and unique addressing scheme for each single management object corresponding to the hierarchical structured arrangement. A specific superordinate object, the root entity, provides the origin of all hierarchically arranged management objects, i.e. the root object of the management tree is that object relative to which the management objects are arranged.

Different kind of management objects are distinguished. The hierarchical object structure, in particular the management tree, comprises interior objects, in the following also denoted as (interior) nodes, and leaf objects. The leaf objects may be further distinguished between leaf objects which are allowed to include any but pre-defined contents and link objects, which include address information of a certain object within the hierarchical object structure. An interior object or an interior node represents an entity used for associating one or several dependent objects thereto, i.e. represents a hierarchically structuring entity having one or several but in a certain manner predetermined number of directly (immediately) associated objects being hierarchically arranged subordinately, which will be denoted as one or more child objects or child nodes being associated with and arranged to a parent object or parent node, respectively. Each interior object or interior node itself is associated to one superordinate interior object or interior node or the root object/node which is the interior object or interior node having the highest hierarchical level. Based on this inter-concatenation the hierarchical object structure, i.e. the management tree, is set up. The interior objects are further distinguished between permanent interior objects and into dynamic interior objects. As the naming "permanent" indicates, permanent objects are always present in the management tree independently from any operational state of the managed client device or application(s) running, whereas the dynamic objects may be added, replaced, modified and/or deleted by the managed client device or the management server corresponding on the need of information being contained in the hierarchical structure (dynamic part of the management tree) being associated with such a dynamic object.

A leaf object is associated to a superordinate interior object but allows no association of subordinate objects, i.e. have no associated child object(s). Instead leaf objects are employed as container for including certain content of any but pre-defined content type, i.e. may include a certain content, value(s) etc. such like integer values, larger values (plain text, array of values), complex data types, pictures and program code. The leaf objects include the configuration data and/or settings required for device and/or application operation. Leaf objects represent the "end"-

objects of branches of the management tree, i.e. the lowest hierarchical level of the respective branches of the management tree.

Fig. 1b shows an example part of a management tree as implemented in a managed client device, e.g. a mobile communication device. The example part of the management tree comprise the root object "/" emerging from which the management tree is set up. Herein the depiction of the example part of the management tree is limited to a highest level structure. Subordinate to the root object, there are depicted a device description framework (DDF) object, a device information (DevInfo) object, a device detail information (DevDetail) object and an example object *AP* which all represent interior objects allowing for arranging subordinately further objects thereto. The number of objects associated to the root object (root node) "/" is not limited to the depicted ones, these are just for example illustration, i.e. non, one or more additional objects may be associated subordinately with the root object "/". The objects DDF, DevInfo and DevDetail represent a selection of mandatory objects (but not limiting thereto) for a managed client device allowing to operate with SyncML device management service. The DevDetail object and the DevDetail object are interior objects which means that an unlimited number of further interior objects and leaf objects (and link objects) are arranged subordinately thereto. The management tree may be set up at a switching on of the managed device clients. Adaptations and/or modifications of the management tree may be operated in accordance with the above described exchange of management information corresponding to the SyncML device management service.

Each object and hence each information contained in a certain object is addressable. The addressing of a certain object is based on a instance identifier. Instance identifiers are assigned to each object of the management tree. At assigning instance identifiers to objects it is to be considered that the addressing of each object has to be unique. That means, an interior object may have one or more directly associated subordinate objects (interior objects, leaf objects, link objects). In order to obtain a unique addressing for each object within the management tree the instance identifiers of these directly associated subordinate objects have to differ to each other. An address information comprises all instance identifiers originating from the root object to the object to be addressed, which are passed by following the hierarchical structure of management tree. A adequate delimiter separates the respective instance identifiers.

The concept of the management tree is provided to a broad number of managed client devices implementing different functions and/or to a managed client devices operation a board number of different application which are not identical, which means that the managed client device are not describable with the same structure of the management tree and consequently the behavior of the managed client devices in view of the SyncML device management service differ.

This problem is addressed by the device description framework (DDF). The usage of the device description framework (DDF) allows a manufacturer of a managed client device for providing information describing the management tree in all its different dynamic (run-time) representations such that the managed client device is capable to be managed by a management server being based on the SyncML device management service. The device description framework (DDF) should be embodied in such a way that it is applicable in a flexible and easily expandable way and also covers future demands to the management tree. The device description framework (DDF) is an abstract description of the objects. Consequently the management tree itself is derivable from the device description framework (DDF), wherein the device description framework (DDF) allows for deriving a broad number of individual management tree structures all valid for use with the corresponding managed client device and adapted to different situations. Therefore, the device description framework (DDF) comprises description objects allowing for deriving objects (permanent/dynamic interior objects, leaf objects and link objects), their properties and their concatenation (association, arrangement within the management tree). The description objects of the device description framework (DDF) include therefore definitions describing in detail information required for setting up the objects and consequently the management tree, respectively, in a valid manner. The description objects of the device description framework (DDF) reflect the hierarchical object structure and acts as a template hierarchical structure of a resulting management tree.

The inventive concept of the present invention relates to rules and regulations which are to be applied during defining and adding of objects described in the device description framework (DDF) such that the above mentioned proclaimed advantages are achieved.

The description of the management tree and the corresponding device description framework (DDF) will be given in reference to graphical depictions.

Fig. 2a shows a table illustrating elements used for graphical notation of object distinguished by their object types. The illustrative representation is compared with the meaning (object type) of the object. Corresponding to the objects available for forming a management tree, the table comprises a fixed (parent) object type, a run-time (parent) object type and a leaf object type. Herein a further object type is presented, a link object type. As aforementioned, the link object type is a specific leaf object type which stores addressing information of a certain different object within the hierarchical object structure, i.e. the link object type defines a leaf object type which includes a link value which is a plain text value to code a uniform resource indicator (URI) or any similar or related addressing value. A fixed parent object having the fixed parent object type

is represented by a rectangular containing a title representing the instance identifier, herein replaced with the word "Fixed", a run-time object having the run-time object type is represented by a rectangular having rounded corners and containing a title, herein replaced with the word "Run-Time", a leaf object having the leaf object type is represented simply by its title representing the instance identifier, herein replaced with the word "Leaf", and finally a link object having the link object type is represented simply by its title being bold styled and underlined and representing the instance identifier, herein replaced with the word "**Link**".

The title (i.e. the instance identifier) of an object is one property element, designated as NodeName element, of the description objects. As aforementioned, an object is identifiable by its NodeName element, which has a predefined value (fixed object) or a dynamic assigned value (run-time object). Further property elements must or may be contained: AccessType, DefaultValue, Description, DFFormat, Occurrence, Scope, DFTitle and DFType. The presented list of property elements is just for example and it is to be understood as not limiting thereto. The AccessType element specifies which commands are allowed on the object (which actions are allowed to be taken against the object) and must be defined. Possible specifications for the AccessType element are Get, Delete, Add and Replace. The DefaultValue element defines optionally the object value to be used in a device unless the object value is not specifically set to a different value. The Description element defines an optional human readable description of the object. The DFFormat element defines a mandatory data format of the object. The Occurrence element specifies optionally the number of instances that may occur of that object within the management tree, i.e. specifically the number of child objects being arranged directly to associated subordinately with one parent object. The Scope element specifies optionally whether this object describes a permanent or dynamic object. The DFTitle element defines an optional human readable name of the object. The DFType element defines for leaf objects the MIME type (multipurpose internet mail extension type) of the object value and for interior objects null or a DDF document name.

The fixed parent object represents a permanent or a dynamic interior object. The title of the parent object is fixed and used as instance identifier in the management tree. In case the Occurrence of the fixed parent object may be one (One), the corresponding interior object is a permanent interior object, i.e. the Scope is permanent, whereas in case the Occurrence of the fixed parent object is none or one (ZeroOrOne) the Scope of the fixed parent object is dynamic such that the interior object can be created and deleted at run-time by a management server.

The run-time (parent) object represents a dynamic interior object. The title of the run-time object is not fixed, i.e. the name of a dynamic interior object is defined dynamically at run-time during

its creation and is used as instance identifier in the management tree. The dynamically defined instance identifier may comprise any sequence of alphanumeric characters (max. 9 characters). In the following the title of a run-time object will be represented by <X> or <Y>.

The titles of the leaf object and the link object are fixed, respectively. The content type of the content included in the leaf object has to be specified, i.e. its MIME type, whereas the content type of the link object specifies a content type of the corresponding leaf object for coding an address information, for example a URI (uniform resource indicator).

In the following rules, regulations and methods presented in conjunction with objects having these certain objects types. The presented object type should not limit the object types thereto, since further object types may be included which do not interfere the inventive concept.

Fig. 2b shows a table illustrating special characters used additionally for graphical notation of objects. The table compares characters and their meanings. The characters represent the Occurrence property element of an object. The Occurrence element specifies optionally the number of object instances that may occur thereof in the management tree. By default and in case no additional character is appended to the title of the object the occurrence of the corresponding object in the management tree is (exactly) one. The "+" sign appended to the title of the object indicates that the occurrence of the corresponding object in the management tree is once or may be also more times than once (Occurrence: OneOrMore, OneOrN). The "*" sign appended to the title of the object indicates that the occurrence of the corresponding object in the management tree is none or may be also more times than none (Occurrence: ZeroORMore, ZeroOrN). The "?" sign appended to the title of the description object indicates that the occurrence of the corresponding object in the management tree is none or once (Occurrence: ZeroOrOne).

The following Fig. 3a to Fig. 3e illustrate rules and regulation to be applied and to be observed during the defining and adding of an object in the hierarchical object structure which will be embodied in the following Fig. 4a by the means of an example method for adding at least one object according to embodiments of the invention.

Fig. 3a shows a first arrangement of description objects according to an embodiment of the invention. This first arrangement includes a first fixed (parent) object "*Node_1*", one or more run-time (parent) objects <X> * and a second fixed (parent) object "*Node_2*". The one or more run-time objects <X> * are associated subordinately with the first fixed (parent) object "*Node_1*", whereas the second fixed parent object "*Node_2*" is associated subordinately with each of the one or more the run-time objects <X> *.

Basing on the above described device description framework (DDF) structure and on the graphical definition of description objects, a resulting part of a management tree may comprises the first fixed parent object "*Node_1*" with the instance identifier "*Node_1*" and none, one or more run-time objects <X> *. The instance identifier(s) of these run-time objects <X> * are defined at run-time and may be for example "run-time_1", "run-time_2" and so on.

It may be assumed that one run-time object <X> * with an instance identifier "run-time_1" has been created (in accordance with the depiction). The run-time object "run-time_1" is associated subordinately with the fixed object "*Node_1*", i.e. the run-time object "run-time_1" is a child object of the fixed object "*Node_1*". In accordance with the scope definition (Scope element) of the second fixed object "*Node_2*", the run-time object "run-time_1" is be parent object to the fixed object "*Node_2*" or the fixed object "*Node_2*" is a child object to the run-time object "run-time_1".

It may be assumed that two run-time objects <X> * with the instance identifiers "run-time_1" and "run-time_2" has been created. The run-time objects "run-time_1" and "run-time_2" are child objects to the fixed object "*Node_1*". And in accordance with the scope definition (Scope element) of the second fixed object "*Node_2*", the run-time object "run-time_1" is a parent object to the fixed object "*Node_2*" and in parallel the run-time object "run-time_2" is analogously a parent object to the fixed object "*Node_2*". The both child objects with the same instance identifier "*Node_2*" are distinguishable via their different dependencies (association, parent objects) within the management tree.

Fig. 3b shows a second arrangement of description objects according to an embodiment of the invention. This second arrangement includes a first fixed object "*Node_3*", one or more run-time objects <Y> +, a second fixed object "*Node_4*" and a leaf object "Leaf". The one or more run-time objects <Y> + are associated subordinately with the first fixed object "*Node_3*", whereas the second fixed object "*Node_4*" and the leaf object "Leaf_1" are associated subordinately with each of the one or more the run-time objects <Y> +. The Fig. 3b illustrates the "simplest" situation, that means only one time object <Y> + which has two child objects, one fixed object "*Node_4*" and one leaf object "Leaf_1".

Basing on the above described device description framework (DDF) structure, a resulting part of a management tree comprises the first fixed object "*Node_3*" with the instance identifier "*Node_3*" and at least one or more run-time objects <Y> + which are directly arranged to and associated subordinately with the first fixed object "*Node_3*", i.e. are child objects of the first

fixed object "*Node_3*". The instance identifier(s) of these child objects are defined at run-time and may be referred analogously as for example "run-time_1", "run-time_2" and so on. Each of the run-time objects "run-time_1", "run-time_2", ... have two child objects, the second fixed object "*Node_4*" and the leaf object "Leaf_1".

The structures described with respect to Fig. 3a and Fig. 3b represent device description framework (DDF) definition structure allowed for use. Contrary thereto, Fig. 3c shows arrangements of objects which are to be avoided in accordance with the rules and regulations of the inventive concept.

The first arrangement of description objects exhibits several first run-time objects <X> * in detail the occurrence of which is none, once or more times and several second run-time objects <Y> * in detail the occurrence of which is also none, once or more times. The depiction of Fig. 3c shows a concatenation of one first run-time object <X> * and one second run-time objects <Y> *. More generally, in case there exists one or more first run-time objects <X> * and one or more second run-time objects <Y> *, each of the first run-time objects <X> * is a parent object to one or several second run-time objects <Y> *, being child objects thereto. Consequently, each of the one or more first run-time objects <X> * being a parent object has a dynamical assigned instance identifier. Moreover, each of the one or more second run-time objects <Y> * being a child object has also a dynamical assigned instance identifier.

It is easy to understand, that such a hierarchical object structure which comprises consecutive directly arranged run-time objects with dynamical assigned instance identifiers in different hierarchical levels exhibits a complex structure which is hard to manage due to the lacking ability to identify objects easily especially in view of exploring the management tree.

As an overall rule for defining objects, a run-time object has to be associated subordinately with a fixed parent object and further this run-time object has to act as a parent object to at least one subordinately associated fixed object and/or a leaf object and/or a link object.

Fig. 3d shows an arrangement of objects to be concentrated to a single object according to an embodiment of the invention. The arrangement comprises two consecutive fixed objects, a fixed object "*Node_5*" and a fixed object "*Node_6*", wherein the only child object of the fixed object "*Node_5*" is the fixed object "*Node_6*". The stringing together of fixed parent objects in the described kind do not exhibit any useful information such that both the fixed object "*Node_5*" and a fixed object "*Node_6*" may be concentrated to a single fixed object "*Node_5Node_6*" which exhibits the same information. The concentration simplifies a resulting management tree

in view of exploring the management tree and accessing object arranged subordinately with the concentrated fixed parent object "*Node_5Node_6*" by simplifying e.g. their addressing.

Fig. 3e shows an arrangement of objects which is to be avoided according to an embodiment of the invention. The arrangement comprises a fixed object "*Node_7*" being associated directly to child objects, a first run-time object <X> * and a second run-time object <Y> +. The first run-time object <X> * and a second run-time object <Y> + each related to a different format of object. That means, the first run-time object <X> * and the second run-time object <Y> + further relate to different functions and applications, respectively. The format of an object shall define which kind of management related information is allowed and/or suitable for being distributed among the respective object and hierarchically arranged objects being associated subordinately with that respective object. Objects having a common format define that similar management related information being associated with the same device function and/or device application is distributed among these objects and subordinately arranged objects.

In order to obtain a manageable and clear management tree different type of run-time objects should not be associated subordinately with a common parent object (being a fixed object) such as shown in Fig. 3e. Rather, run-time objects of different format relating to different functions and applications of the managed client device, respectively, are to be associated to different parent objects (being a fixed object) to provide a clearly systematic, useable and manageable management tree.

Fig. 4a shows a first flow chart illustrating the defining of an object in accordance with an embodiment of the invention. The illustrated first flow chart represents an example embodiment taking into account the above illustrated and described rules and regulations, respectively, for defining objects and their properties and dependencies.

In an operation S100, the operational sequence according to an embodiment of the present invention for defining/adding at least one new object is started.

In an operation S110, the object type of the parent object is determined. The parent object to a new object can be a fixed object or a run-time object. In accordance to the rule and regulation described with reference to Fig. 3c, it is to be avoided that two run-time objects are arranged consecutively in the hierarchical object structure and device document framework (DDF), respectively. Correspondingly, it is checked whether the parent object is a run-time object or a fixed object. In case the parent object is of a run-time type the depicted operational sequence is continued with operation S140, ensuring that no consecutive run-time object can be defined and

added. In case the parent object is of the fixed type the depicted operational sequence is continued with operation S120 allowing for associating any type of object to the parent object being a fixed object.

In an operation S120, it is checked whether a new run-time object is to be defined/added and in case it is the operational sequence continues with operation S130. Otherwise the operational sequence continues with operation S140.

In an operation S130, the new run-time object is defined. The defining involves an associating (or adding or including) of the new run-time object to the superordinately arranged parent object and the hierarchical object structure, respectively. In correspondence with the above described rules and regulations the superordinately arranged parent object is a fixed object. The defining of the new run-time object further may include the defining of further property elements comprising AccessType, DefaultValue, Description, DFFormat, Occurrence, Scope, DFTitle and DFType. After a complete defining of all necessary and required property elements and the adding of the new run-time object into the hierarchical object structure the operational sequence for defining a new object is continued with operation S200, where the operational sequence ends. Further new objects may be defined by starting again the described operational sequence.

In an operation S140, it is checked whether a new fixed object is to be defined and in case it is the operational sequence continues with operation S150. Otherwise the operational sequence continues with operation S160.

In an operation S150, the new fixed object is defined. The defining involves an associating (or adding or including) of the new fixed parent object to the superordinately arranged parent object and the hierarchical object structure, respectively. In correspondence with the above described rules and regulations the superordinately arranged parent object is either a fixed parent object or a run-time (parent) object. The defining of the new fixed parent object further may include the defining of further property elements comprising AccessType, DefaultValue, Description, DFFormat, Occurrence, Scope, DFTitle and DFType. After a complete defining of all necessary and required property elements and the adding of the new fixed parent object into the hierarchical object structure the operational sequence for defining a new object is continued with operation S200, where the operational sequence ends. Further new objects may be defined by starting again the described operational sequence.

In an operation S160, it is checked whether a new leaf object is to be defined and in case it is the operational sequence continues with operation S170. Otherwise the operational sequence continues with operation S180.

In an operation S170, the new leaf object is defined. The defining involves an associating (or adding or including) of the new leaf object to the superordinately arranged parent object and the hierarchical object structure, respectively. In correspondence with the above described rules and regulations the superordinately arranged parent object is either a fixed parent object or a run-time (parent) object. The defining of the new leaf object further may include the defining of further property elements comprising AccessType, DefaultValue, Description, DFFormat, Occurrence, Scope, DFTitle and DFType. After a complete defining of all necessary and required property elements and the adding of the new leaf object into the hierarchical object structure the operational sequence for defining a new object is continued with operation S200, where the operational sequence ends. Further new objects may be defined by starting again the described operational sequence.

In an operation S180, it is checked whether a new link object is to be defined and in case it is the operational sequence continues with operation S180. In the present invention four different types of objects are presented. Correspondingly, the checking operations S120, S140, S160 and S180 cover this four types. It should be noted that one or more further types of objects may be included into the device document framework (DDF). In accordance with the current situation in case new link object is not to be defined the operational sequence is continued with operation S200, i.e. the operational sequence ends without any new object definition. The operations of checking, defining and adding for one or more further types of objects may be implemented in the present operational sequence analogously to the checking, defining and defining operations described above with reference to the respective types of objects.

In an operation S190, the new link object is defined. The defining involves an associating(or adding or including) of the new link object to the superordinately arranged parent object and the hierarchical object structure, respectively. In correspondence with the above described rules and regulations the superordinately arranged parent object is either a fixed parent object or a run-time (parent) object. The defining of the new link object further may include the defining of further property elements comprising AccessType, DefaultValue, Description, DFFormat, Occurrence, Scope, DFTitle and DFType. After a complete defining of all necessary and required property elements and the adding of the new link object into the hierarchical object structure the operational sequence for defining a new object is continued with operation S200, where the

operational sequence ends. Further new objects may be defined by starting again the described operational sequence.

In an operation S200, the operational sequence according to an embodiment of the present invention for defining/adding a new object is finished.

It shall be noted that alternatively to the presented operational sequence which ensures that run-time objects are not arranged consecutively in a hierarchical object structure, an operation of adding and interposing of a fixed object within two consecutively arranged run-time objects also fulfills this regulation. The interposing of a fixed object within the consecutive run-time objects prevents this consecutive arrangement and leads finally to the same resulting hierarchical object structure.

According to a further (improved) embodiment of the invention, the operational sequence also takes into account a checking of the object formats of run-time objects being already associated to the parent object to which the new object is to be directly associated as an additional client object. The corresponding rule and regulation is described in detail with reference to Fig. 3e, respectively.

In the operation S130, the existing "parallel" child objects of the parent object are determined. In case there are existing description child objects, it should be avoided that run-time objects of different formats are associated to the same parent object. In case a run-time object of different format is to be defined to be associated in parallel to an existing run-time object having a certain format the defining rejected and the operational sequence is continued with operation S200, i.e. the operational sequence ends.

According to a further (improved) embodiment of the invention, the operational sequence also takes into account a checking of the object types being already associated to the parent object to which the new object is to be directly associated as a/an (additional) client object. The corresponding rule and regulation is described in detail with reference to Fig. 3d, respectively.

In the operation S150, the existing "parallel" child objects of the parent object are determined. In case no "parallel" description child objects exists and the parent object is a fixed object, the new fixed object and the existing superordinately associated fixed parent object are concentrated into a new single fixed object replacing the parent object in order to simplify the hierarchical object structure and the device description framework (DDF), respectively.

Fig. 4b shows a second flow chart illustrating a defining and creating, respectively, of at least a part of a device description framework (DDF) and DDF document in accordance with an embodiment of the invention. This operational sequence involves the operational sequence depicted in Fig. 4a according to an embodiment of the present invention.

In an operation S300, the operational sequence for creating a DDF document in accordance with the defining of new objects according to an embodiment of the invention is started.

In an operation S310, an object is defined. By default, the root object is existent, i.e. being the basis relative to which the device description framework (DDF) is structured and consequently the DDF document is built-up. The defining of an object in the operation S310 comprises the operations depicted in Fig. 4a and described in detail with reference thereto.

In an operation S320, the resulting defined new object is coded in the DDF document in accordance with the definitions having been performed in the operation S310. The coding of the DDF document may base on extended markup language (XML) coding. The XML coding is based on a type description provided in a corresponding document type description (DTD) defining the language elements of the XML encoding required for the DDF document. The XML encoding may be any suitable XML encoding such as a binary coded XML encoding.

In an operation S330, in case a further new object is to be added to the current device description framework and the DDF document, respectively, the operational sequence returns to the operation S310. Otherwise in case the current device description framework and the DDF document is complete, respectively, i.e. no further object is to be added thereto, the operational sequence continues with operation S340.

In an operation S340, the resulting device description framework and DDF document is stored in or may be transmitted to a management server and/or a managed client device, respectively, to be applied for establishing, generating, modifying etc. of a management tree. Moreover, the resulting device description framework and DDF document is processed for generating at least a part of a management tree, respectively. Such a processing of a device description framework and DDF document will be described below with reference to Fig. 4c, respectively.

In an operation S350, the operational sequence for creating a DDF document in accordance with the defining of new objects according to an embodiment of the invention is finished.

It shall be noted, that the operation S310 of defining a new object and the operation S320 of coding the new object has been described as separated operations. It is to be understood, that both the operation S310 and the operation S320 may be embodied as a comprehensive operation, such that the defining and coding are operated simultaneously.

Fig. 4c shows a third flow chart illustrating the parsing and generating of at least a part of a management tree being based on a device description framework (DDF) and a DDF document, respectively, in accordance with an embodiment of the invention.

In an operation S400, the operational sequence for creating/generating at least a pert of a management tree being based on a device description framework (DDF) and a DDF document according to an embodiment of the invention is started, respectively.

In an operation S410, the device description framework (DDF) and the DDF document comprising the coded objects in correspondence to which objects of a management tree are to be generated is retrieved from a storage or may be received from a providing entity such as a device management server or a supporting server of the manufacturer of the electronic device, . For example, a device description framework (DDF) and a DDF document may be received by a managed client device from a management server or may be received by a management server from a DDF providing networked server. As aforementioned, the availability of the device description framework (DDF) and the DDF document, respectively, allows a processing device to generate valid and appropriate management tree structures and guaranties their applicability.

In an operation S420, the device description framework (DDF) and the DDF document, respectively, is parsed. The parsing may comprise an extracting of information and/or an interpreting of the information obtained by parsing. The parsing may be operated object by object as embodied herein or may be operated block-wise, i.e. may be operated by parsing a set of related description objects. The parsing of the device description framework (DDF) and the DDF document may further comprise an identifying operation, respectively, in order to parse/extract the required object(s) information from the total device description framework (DDF) and the DDF document, respectively.

In an operation S430, based on the information obtained by operation S420, one or several objects are generated. The generation of each object is performed in correspondence with the object information comprised in the device description framework (DDF) and the DDF document, respectively. The generated objects are included in the at least part of the management tree to be generated. The including of the new objects is performed under consideration of the

hierarchical association defined in the device description framework (DDF) and the DDF document, respectively, in order to generate the at least part of a management tree with the correct hierarchical structure. Moreover the method for adding an object to a hierarchical object structure as embodiment with reference to Fig. 4a may be applied to ensure that (even in case the DDF document has not taken the aforementioned rules and regulation into consideration) the resulting hierarchical object tree (the management tree) meet the inventive rules and regulations.

If necessary, in the operation S430 one or more values are further assigned to the new generated object(s).

In an operation S440, in case the object information required is parsed/extracted object by object and the generation of the at least part of the management tree has not been completed, the operational sequence returns to the operation S420 for parsing/extracting a further required object information. Otherwise the generation of the at least part of the management tree is completed and the operational sequence continues with operation S450.

In an operation S450, the resulting at least part of the management tree generated is applied. The applying of the resulting at least part of the management tree may be an establishment of a management tree in a managed client device, an implementation of an additional branch into an existing management tree for adding additional configuration data or settings to the management tree required by one or more device functions and/or applications. Moreover, the resulting at least part of the management tree may have been generated by a management server and transmitted to a managed client to be implemented therein.

The applying of the resulting at least part of the management tree should been understood in conjunction with the above described SyncML device management service and the exchange of a management document.

In an operation S460, the operational sequence for creating/generating at least a part of a management tree in accordance with a device description framework (DDF) and a DDF document according to an embodiment of the invention is finished, respectively.

The following Fig. 5a will show a graphical depiction of an example part of a device description framework. The Fig. 5b and Fig. 5c show the corresponding DDF document and the Fig. 5d illustrate an example management tree which is in agreement with the example device description framework illustrated in Fig. 5a and coded in form of a document shown in Fig. 5b and Fig. 5c..

Fig. 5a shows an example part of a graphical depiction of a device description framework according to an embodiment of the invention. The graphical depiction relates to the access point (AP) settings of a mobile data communication enabled client device. The graphical depiction shows a part of the relevant objects being based on that one or more branches of the management tree of the mobile data communication enabled client device are derived.

The access point settings are sub-summarized below a highest level interior object "./AP" directly associated to the root object "/", i.e. the object "./AP" is a child object of the root object "/" and is a parent object to all access point (AP) settings. Correspondingly, the depicted highest level object "./AP" defines a fixed parent object with the instant identifier "./AP". The fixed parent object "./AP" is associated to a run-time object $<X_1>$ * arranged subordinately thereto. The run-time object $<X_1>$ * allows to derive none, one or more corresponding interior objects with dynamically assigned instant identifiers being child objects of the corresponding interior parent objects "./AP". Each set of information relating to individual access point settings is stored in one of the management tree substructure below one of the run-time objects $<X_1>$ *. The instant identifier of these run-time objects $<X_1>$ * indicate preferably the kind of access point to which the individual access point settings relate.

Each one of the run-time objects $<X_1>$ * is associated with a fixed object "Px", a fixed object "NAPDef ?", a leaf object "ClientID ?" and a fixed object "BS ?". These objects are allowed to occur not or exactly once in the management tree being arranged subordinately to each one of the run-time objects $<X_1>$ *.

The fixed object "Px" is parent object to one or more run-time objects $<X_2>$ *, the fixed object "NAPDef ?" is parent object to one or more run-time objects $<X_3>$ * and the fixed object "BS ?" is parent object to a run-time object $<X_4>$ *. Assuming that the fixed object "Px" is existing in the management tree the fixed object "Px" may have none, one or more run-time objects $<X_3>$ *with dynamically defined instant identifiers. Analogously on assumption that a fixed object "NAPDef ?" is existing in the management tree the fixed object "NAPDef ?" may have none, one or more run-time objects $<X_3>$ * and on assumption that the fixed object "BS ?" is existing in the management tree the fixed object "BS ?" may have none, one or more run-time objects $<X_4>$ *. The further structures of the run-time objects $<X_2>$ * and $<X_3>$ * are omitted.

The run-time object $<X_4>$ * has several child objects comprising among other for example a leaf object "Name ?", a fixed parent object "Network ?" and a leaf object "Country ?".

In turn, the fixed object "Network ?" is associated to one or more run-time object <X5> + being arranged subordinately thereto. According to the definition of the run-time object <X5> + in case the fixed object "Network ?" is existing at least one or more run-time object $<X_5>$ + are associated to the fixed object "Network ?" as interior child objects.

It can be seen, that run-time objects in the depicted structure are always separated by at least one fixed object and that a consecutive arranging of fixed objects has been avoided and that not different formats of run-time objects coexist as child objects of one superordinate (fixed/run-time object) parent object such that the depiction fulfills the above described rules and regulations, respectively.

The graphical depiction of the device description framework may be employed for obtaining a corresponding DDF document. The meanings of the graphical elements comprised by the graphical depiction may be translated into the corresponding DDF document to at least obtain a framework of the DDF document.

Fig. 5b shows a first part of an excerpt of a DDF document corresponding to the graphical depiction shown in Fig. 5a. Fig. 5c shows a second part of the excerpt of the DDF document shown in Fig. 5b. In the following, the Fig. 5b and Fig. 5c will be described together.

The following DDF document excerpts are based on an extended markup language (XML) encoding of the device description framework. The XML encoding is moreover based on a document type description defining tags for defining the objects and their properties in a manner such that the resulting DDF document is interpretable in a unique way. The XML encoding is one of a board number of possible encoding methods. The following DDF document is based on a document type description for device description framework provided by the SyncML Initiative.

Lines 001 to 007 include the header section of the DDF document. The header sections defines the XML encoding version (1.0), a character encoding (UTF-8), the version of the DTD to be considered (1.1), a manufacturer (Nokia) a model identification (omitted by comment) of the client device to which the DDF document relates.

In lines 008 to 018 the part of the XML encoded DDF document is comprises that is dedicated to the fixed object "AP", its position in the management tree relative to the root object "/" and the property elements of the fixed object "AP". In correspondence with the aforementioned properties of the fixed object "AP", a set of object property elements specify the properties

thereof. In detail, the access is limited to read operations, the format of the object is set to node indicating an interior object, the occurrence is defined as one and the scope is permanent, Additionally, a human readable description and a human readable title are defined.

In lines 019 to 026 the run-time object <X_1> * is defined. A name is omitted, since the instance identifier of the run-time object is assigned at run-time. The object property elements specify the access to add, delete, get and replace, the format of the object is defined as node, the occurrence is none, one or more (ZeroOrMore) and the scope is dynamic. A human readable title is defined.

In lines 027 to 037 the fixed parent object "Px" is specified. The name is correspondingly defined as "Px". The object property elements specify the access limited to read operations (get), the format of the object is defined as node, the occurrence is one and the scope is dynamic. A human readable title is defined.

In line 036 omission marks indicate the position in the depicted DDF document at which the specification of the run-time object <X_2> * is to be included. The specification of the run-time object <X_2> * is performed analogously to the specification of run-time objects presented herein.

In lines 038 to 048 the fixed object "NAPDef?" is specified. The name is correspondingly defined as "NAPDef". The object property elements specify the access limited to read operations (get), the format of the object is defined as node, the occurrence is none or one and the scope is dynamic. A human readable title is defined.

In line 047 omission marks indicate the position in the depicted DDF document at which the specification of the run-time object <X_3> * is to be included. The specification of the run-time object <X_3> * is performed analogously to the specification of run-time objects presented herein.

In lines 049 to 059 the leaf object "ClientID ?" is specified. The name is correspondingly defined as "ClientID". The object property elements specify the access to add, delete, get and replace, the format of the object is defined as character format (chr) being a certain leaf object, the occurrence is none or one and the scope is dynamic. A human readable title is defined. The leaf object contains information. The kind of information, i.e. the content type format of the data representing the information, has to be predefined. Therefore, the corresponding type element (DFType) comprises a MIME type definition of the content to be stored, herein a plain text information.

In lines 060 to 069 the fixed parent object "BS ?" is specified. The name is correspondingly defined as "BS". The object property elements specify the access limited to read operations (get), the format of the object is defined as node, the occurrence is none or one and the scope is dynamic. A human readable title is defined.

In lines 070 to 077 the run-time object <X_5> * is defined. A name is omitted, since the instance identifier of the run-time object is assigned at run-time. The object property elements specify the access to add, delete, get and replace, the format of the object is defined as node, the occurrence is none, one or more (ZeroOrMore) and the scope is dynamic. A human readable title is defined.

The lines 078 to 088 the leaf object "Name ?" is specified. The name is correspondingly defined as "Name". The object property elements specify the access to add, delete, get and replace, the format of the object is defined as character format (chr) being a certain leaf object, the occurrence is none or one and the scope is dynamic. A human readable title is defined. The content type is specified by a MIME type definition as being plain text.

The lines 089 to 100 the fixed parent object "Network ?" is specified. The name is correspondingly defined as "Network". The object property elements specify the access limited to read operations (get), the format of the object is defined as node, the occurrence is none or one and the scope is dynamic. A human readable title is defined.

In line 098 omission marks indicate the position in the depicted DDF document at which the specification of the run-time object <X_5> * is to be included. The specification of the run-time object <X_5> * is performed analogously to the specification of run-time objects presented herein.

The lines 101 to 110 the leaf object "Country ?" is specified. The name is correspondingly defined as "Country". The object property elements specify the access to add, delete, get and replace, the format of the object is defined as character format (chr) being a certain leaf object, the occurrence is none or one and the scope is dynamic. A human readable title is defined. The content type is specified by a MIME type definition as being plain text.

In line 111 omission marks indicate the position in the depicted DDF document at which further objects associated to the run-time object <X_5> * are to be included.

The hierarchical structure of the device description framework shown accordingly by its graphical depiction in Fig. 5a is mapped to the device description framework and the DDF document, respectively, by encapsulating of the specifications presented and described above.

Due to the employment of the device description framework and the DDF document, respectively, the hierarchical structure being realized by the encapsulations of the object specification represents a hierarchical template structure of the management tree being derived therefrom. Alternatively, instead of using encapsulating of specifications for defining the hierarchical structure, a path information may be added to object specifications. The path information specifies the position within the hierarchical structure of the corresponding object.

The specification range of the fixed object "./AP" covers the object specifications of subordinately arranged objects beginning in line 007 and ending with line 115 wherein the specification of the fixed object "./AP" itself is comprised in this encapsulation.

The specification range of the run-time object <X_1>* covers the object specifications of subordinately arranged objects beginning in line 018 and ending with line 114 wherein the specification of the run-time object <X_1>* itself is comprised in this encapsulation.

The specification range of the fixed object "Px" covers the object specifications of subordinately arranged objects beginning in line 027 and ending with line 037 wherein the specification of the fixed object "Px" itself is comprised in this encapsulation and the omission marks in line 036 indicate omitted object specifications being hierarchically arranged subordinate.

The specification range of the fixed object "NAPDef?" covers the object specifications of subordinately arranged objects beginning in line 038 and ending with line 048 wherein the specification of the fixed object "NAPDef?" itself is comprised in this encapsulation and the omission marks in line 047 indicate omitted object specifications being hierarchically arranged subordinate.

The specification range of the fixed object "BS?" covers the object specifications of subordinately arranged objects beginning in line 060 and ending with line 113 wherein the specification of the parent object "BS?" itself is comprised in this encapsulation.

The specification range of the run-time object <X_4>* covers the object specifications of subordinately arranged objects beginning in line 069 and ending with line 112 wherein the specification of the run-time object <X_4>* itself is comprised in this encapsulation and the omission marks in line 111 indicate omitted object specifications being hierarchically arranged subordinate.

The specification range of the fixed object "Network ?" covers the object specifications of subordinately arranged objects beginning in line 089 and ending with line 099 wherein the specification of the fixed object "Network ?" itself is comprised in this encapsulation.

Fig. 5d shows an example excerpt of a graphical depiction of a management tree in accordance with the graphical depiction of a device description framework illustrated in Fig. 5a according to an embodiment of the invention. The presented management tree is constituted from a plurality of objects being derived from the corresponding objects and being associated with each other in correspondence with the defined hierarchical structure described by the device description framework.

The fixed object "./AP" represents in this depiction the object of the highest hierarchical level, i.e. the object "./AP" is directly arranged to and subordinately associated with the root object "/" of the management tree. The object "./AP" shall comprise management related information concerning the network access of an mobile communication terminal.

The object "./AP" has three child objects, i.e. objects which are directly arranged thereto and subordinately associated therewith, the object "Prov_1", object "Prov_2", object "Prov_3" and further objects. These objects are run-time objects corresponding to the aforementioned run-time object $<X_1>$ *. In accordance with the aforementioned regulation for arranging run-time objects being child objects of a common parent, all these run-time objects have to serve for the same subordinate structure. The real subordinate structure of the run-time objects may differ, since directly and indirectly arranged and subordinately associated objects may be dynamical objects. But the run-time object "Prov_1", object "Prov_2" and object "Prov_3" are set up in such a way that they relate to the same functional or applicational management configuration information. Herein, the run-time object "Prov_1", object "Prov_2" and object "Prov_3" relate to configuration information of network service providers. The fixed object "./AP" serves to sub-summarize all management information concerning network service provider configuration information.

The run-time object "Prov_1" has two child objects, the object "Px" and the object "NAPDef".. In accordance with the aforementioned regulation for consecutively arranging run-time objects, the object "Px" and the object "NAPDef" are both fixed objects . Corresponding to the occurrences defined for further possible directly arranged and subordinately associated objects, a leaf object "ClientID" and a fixed object "BS" may be dynamically arranged as child objects to the object "Prov_1" but must not. Herein, none of these possible further objects are implemented in the management sub-tree of the object "Prov_1". The object "Px" relates to proxy

configuration information of the network provider service "Prov_1" and the object "NAPDef" relates to network access point definition configuration information of the network provider service "Prov_1".

The fixed object "Px" being arranged subordinately to object "Prov_1" has two child objects, a run-time object "Def_Px" and a run-time object "Px_1". Both the run-time object "Def_Px" and the run-time object "Px_1" correspond tp run-time object $<X_2>$ *. The object "Px" serves to sub-summarize all proxy configuration information of the network provider service "Prov_1". Further run-time objects are only allowed to be associated as child objects to the object "Px" if these further run-time objects also correspond to the definitions described under consideration of the run-time object $<X_2>$ *, i.e. have the same (and a common, respectively) format.

The fixed object "NAPDef" being arranged subordinately to object "Prov_1" has two child objects, a run-time object "NAP_Def" and a run-time object "NAP_1". Both the run-time object "NAP_Def" and the run-time object "NAP_1" correspond to the run-time object $<X_3>$ *. The object "NAPDef" serves to sub-summarize all network access point definition configuration information of the network provider service "Prov_1". Further run-time objects are only allowed to be associated as child objects to the object "NAPDef" if these further run-time objects also correspond to the definitions provided by the run-time object $<X_3>$ *, i.e. have the same (and a common, respectively) format.

In case of the run-time objects "Def_Px" and "Px_1" the fixed object "Px" separates these run-time objects from the superordinately arranged run-time object "Prov_1" which corresponds to the aforementioned regulation for separating run-time objects. Analogously, in case of the run-time objects "NAP_Def" and "NAP_1" the fixed object "NAPDef" separates these run-time objects from the superordinately arranged run-time object "Prov_1" which corresponds to the aforementioned regulation for separating run-time objects.

The fixed object "Prov_2" has three child objects, a fixed object "NAPDef", a leaf object "ClientID" and a fixed object "BS". As the directly arranged and superordinately associated object "Prov_2" is of the run-time type, the parent object "NAPDef" as well as the parent object "BS" have both the fixed object type.

The fixed parent object "NAPDef" being arranged subordinately to object "Prov_2" has one child object, a run-time object "NAP_Def".

The fixed object "BS" being arranged subordinately to object "Prov_2" has one child object, a run-time parent object "Boot". The object "BS" relates to bootstrap configuration information of the network provider service "Prov_2". In correspondence with the object $<X_3>$ * the object "BS" has the child objects "Name" and "Country" both having leaf object type. Further child objects corresponding to the object $<X_3>$ * may be associated to the object "BS".

Fig. 6 shows a block diagram illustrating devices containing components for operating the aforementioned methods according to embodiments of the invention. A server device management agent 220 represents a networked service that provides device management with another counterpart client device management agent 320. The device management data may be provided or processed by the server device management agent 220 or client device management agent 320, respectively. The server device management agent 220 is hosted by the server 20 which may be a server device corresponding with the server device mentioned with reference to Fig. 1. Analogously, the client device management agent 320 is hosted by the client 30 which may be a client device corresponding with the client device mentioned with reference to Fig. 1. The device management is performed between a server 20 and a client 30.

The server 20 and client 30 are connected over any network. The network provides a logical communication connection between the server 20 and client 30, allowing the establishment of the end-to-end communication during the device management which may be termed as device management session. A selection of logical connections and bearers thereof are described in Fig. 1.

The client 30 may use the client device management agent 320 in conjunction with the synchronization adapter 340 and synchronization interface 330 to access the network and send messages to the server via the synchronization adapter 340 and synchronization interface 330 in accordance to the SyncML DM protocol standard. The server 20 or server device management 220, respectively, receives or sends messages via the synchronization adapter 240 and synchronization interface 230, and manages the entire device management process through the server device management engine 210. Device management operations are conceptually bound into a device management frame, which is a conceptual frame for one or more required packages.

The server device management engine 210 has the possibility to access an adapted device management database 200 containing information about the client 30 to be managed such as configuration data and settings relating to certain client device functions or applications operable with the client 30 which are to be transmitted to the client 30 for allowing a user to use well-configured device functions and/or device applications. The device management database 200

may further contain the client related device description framework (DDF) information such as a DDF document defined and provided by the manufacturer for deriving at least a part of the management tree valid for the client device 30, a part of the management tree itself, information about the actual position within the management tree of the client 30 to be processed and further device management relevant information. Further, the server device management engine 210 of the server 20 is able to generate the device management documents exchanged with the client 30. This generation is possible since the DDF information (DDF document) enables the server 20 to code device specific management documents required for being exchanged with the client 30 in an appropriate manner such that the management related information are conform with the management tree implemented in the client 30. For example in case certain management related information shall be transmitted to the client for enabling a certain client function. This management related information shall further be distributed among a certain management tree sub-area (a certain branch) which is at first to be set up dynamically. The server 20 as well as the client 30 are known about the device description framework (DDF) 310 such that the dynamical extension of the management tree is operated in an adequate way which enables the server 20 to instruct the client the distribute transmitted management retailed information among the correct objects of the specific dynamic management tree sub-area and which enables the client 30 to manage and retrieve the dynamical implemented in a correct way which includes providing of this information to the corresponding device function and/or device application requiring it.

The counterpart client 30 is able to response to the management request employing the client device management agent 320. Especially, the client device management agent 320 has access to its device management tree 300 and its device description framework (DDF) 310 defining the hierarchical structure and objects of the management tree 300.

Both, the server 20 and the client 30 may use the DDF information (DDF document) stored therein in order to code action to be performed against management tree of the client 20. The DDF information (DDF document) allows to generate dynamic part(s) of the management tree required for storing new or dynamic management related information concerning the operation of the client 30. The generating of the part(s) of the management tree being based on the DDF information (DDF document) is operated by the server device management agent 220 and the client device management agent 320, respectively, depending in which device the modifications or adaptations of the management tree 300 are operated.

Primarily, the device description framework (DDF) is supplied by a manufacturer of a certain client device such as client 30 to a device management server such as server 20, such that the server is capable to operate the device management with this client by generating management

actions such as described above in a client device appropriate way. Moreover, the DDF information (DDF document) being available to the client 30 allows the client for generating a management tree at switching on of the client 30 in case that no management tree is available for the client 30 up to now. The DDF information (DDF document) serves as mentioned above as a template representing and describing the management tree and the possible management trees in a common, extensible and flexible way, respectively.

The presented components of the server 20 or the client 30, respectively, the server device management agent 220, the server device management engine 210 and the device database 200 respectively, as well as the client device management agent 320 and the device management tree 200, respectively, may be constituted by a data processing device which may be comprised by the server 20 or the client 30, respectively. Further these components may be constituted by a code section for executing on the server 20 or the client 30, respectively, containing instructions for carrying out the necessary processing operations.

It will be obvious for those skilled in the art that as the technology advances, the inventive concept can be implemented in a broad number of ways. The invention and its embodiments are thus not limited to the examples described above but may vary within the scope of the claims.

Claims

1. Method for adding at least one object into a hierarchical object structure, said hierarchical object structure comprising a plurality of objects being hierarchically associated,
wherein said plurality of objects comprises at least two different types of objects allowed to have subordinately arranged objects, called first and second types;
wherein said method comprises:
   - defining said at least one object, called child object, to be associated to one of said objects, called parent object, which is directly arranged to be superordinate to said child object and is part of said hierarchical object structure by:
   - comparing types of said parent object and said child object;

   wherein in case said types of said child object and said parent object are both found to be of said first type, an object of said second type is added between said parent object and said child object.

2. Method according to claim 1, wherein said at least two different types of objects are a run-time type and a fixed type, wherein said type of said first object is said run-time type and said type of said second object is said fixed type.

3. Method according to claim 1 or claim 2, comprising:
   - determining whether said parent object has already one or more directly arranged objects being associated subordinately therewith;
   - in case there are one or more already existing objects, comparing said types of said one or more already existing objects and said child object; and
   - in case said types of said one or more already existing objects and said child object are found to be of said first type, adding said child object having a common format with said matching one or more further objects.

4. Method according to any one of the preceding claims, comprising:
   - determining said type of said parent object; and
   - determining whether said parent object has already existing one or more directly arranged objects being associated subordinately therewith:
   - in case said parent object has none already existing objects and said parent object and said child object are both found to be of said second type:
   concentrating said parent object and said child object by adding a concentrated object being obtained from said parent object and said child object and having said second type.

5. Method according to any one of the preceding claims, comprising:
   - coding at least a part of a description document being based on said definition of said at least one object and comprising information relating to said at least one object and said properties of said at least one object.

6. Method according to any one of the preceding claim, wherein said hierarchical object structure constituted by said plurality of objects describes a hierarchical management structure;
wherein said hierarchical management structure is employed for distributing management related information of a mobile communication enabled device among said plurality of objects, certain parts of said management related information being assigned to at least one of said plurality of objects.

7. Method according to any one of the preceding claim, wherein an object format of an object specifies which kind of management related information is distributed among said object and hierarchically subordinated objects associated thereto.

8. Method according to any one of the preceding claims, wherein said hierarchical object structure is an information being part of the device description framework and/or said hierarchical structure constituted by a plurality of objects is a management tree, both being employed for device management according to the synchronization markup language device management (SyncML DM) standard defined by the SyncML Initiative.

9. Method according to any one of the preceding claims, wherein said description document is at least a part of a device description framework (DDF) document and/or said device description framework (DDF) document being an extended markup language (XML) encoded document being encoded in accordance with a corresponding description framework document type description (DTD).

10. Software tool for adding an entity into a hierarchical structure consisting of a plurality of entities, comprising program portions for carrying out the operations of any one of the claims 1 to 9, when said program is implemented in a computer program for being executed on a processing device, a networked device, a networked server, a terminal device or a communication terminal device.

11. Computer program product for adding an entity into a hierarchical structure consisting of a plurality of entities, comprising loadable program code sections for carrying out the operations of any one of the claims 1 to 9, when said computer program is executed on a processing device, a networked device, a networked server, a terminal device or a communication terminal device.

12. Computer program product for adding an entity into a hierarchical structure consisting of a plurality of entities, wherein said computer program product is comprising program code sections stored on a computer readable medium for carrying out the method of any one of the claims 1 to 9, when said computer program product is executed on a processing device, a networked device, a networked server, a terminal device or a communication terminal device.

13. Computer data signal embodied in a carrier wave and representing a program which, when executed by a processor, causes the method of any one of claims 1 to 9 to be carried out.

14. Processing device having a processing unit, a memory unit and a communication interface, said processing unit being interconnected with said memory unit and said communication interface, wherein said processing unit is configured for defining at least one object to be included into a hierarchical object structure, said hierarchical object structure being constituted by a plurality of objects being hierarchically associated, wherein said plurality of objects comprises at least two different types of objects allowed to have subordinately arranged objects, called first and second objects;
wherein said processing unit is configured for:
- defining said at least one object, called child object, to be associated to one of said objects, called parent object, which is directly arranged to be superordinate to said child object and is part of said hierarchical object structure by:
  - comparing types of said parent object and said child object:

in case said types of said child object and said parent object are both found to be said first type, object of said second type is added between said parent object and said child object.

15. Management system comprising a processing device and a hierarchical object structure, said hierarchical object structure being constituted by a plurality of objects being hierarchically associated;
wherein said plurality of objects comprises at least two different types of objects allowed to have subordinately arranged objects, called first and second objects;

wherein each two objects of said plurality of objects both having the first type are separated by at least one object having the second type independently of their hierarchical position in said hierarchical object structure;

wherein said processing device has at least one management component, wherein said hierarchical object structure is for managing management related information of a managed mobile communication enabled device and wherein said management component is at least capable to generate at least a part of a hierarchical object structure comprising a plurality of objects.

16. Management system according to claim 15, wherein said at least two different types of objects are a run-time type and a fixed type, wherein said type of said first object is said run-time type and said type of said second object is said fixed type.

17. Management system according to claim 15, wherein two or more objects which have said first type and which are directly arranged to an object being associated superordinately therewith have a common format.

18. Management system according to claim 15or claim 16, wherein two directly arranged objects being associated with each other and having both said second type are constituted by a concentrated object having said second type, wherein said concentrated object is constructed from said both directly arranged objects being associated with each other.

19. Management system according to any one of the claims 15 to 17, wherein said processing device is a managed mobile communication enabled device comprising at least a device management component and a storage component, wherein said device management component allows for

- generating said at least a part of said hierarchical object structure to store said generated part of said hierarchical object structure in said storage component, to establish said part of said hierarchical structure;
- distributing management object related information among said plurality of objects constituting said hierarchical object structure; and
- retrieving at least parts of said management related information from one or more objects of said plurality of objects for configuring functions of said managed mobile communication enabled device or for configuring applications operable with said managed mobile communication enabled device.

20. Management system according to any one of the claims 15 to 18, wherein said processing device is a management server able to provide management related information to a managed mobile communication enabled device; wherein said management server comprises at least a device management component and a communication interface for communicating to said managed mobile communication enabled device; wherein said device management component allows for
    - generating said at least part of said hierarchical object structure;
    - generating one or more management messages to transmit management related instructions in accordance with said at least part of said hierarchical object structure;
    - and transmitting said one or more management messages in cooperating with said communication interface to said managed mobile communication enabled device.

Abstract

The present invention disclose an advantageous method for defining object to be included into a hierarchical structured device description framework (DDF). The device description framework (DDF) is constituted by a plurality of objects being hierarchically inter-associated to each other forming a kind of tree-like structure. The hierarchical structure is obtained by using different types of object comprising at least: fixed parent object, run-time parent object and leaf object. Each object has one directly arranged and superordinately associated object, i.e. a parent object, whereas objects of the types fixed parent object and run-time parent object are allowed for having none one or more objects being directly associated subordinate, i.e. child objects. At least one new object is defined to be associated to a parent object. The object type of the parent object is checked. In case the parent object is a fixed parent object, the at least one new object is allowed to be either fixed object, run-time object or leaf object. In case the parent object is a run-time object, the at least one new object is allowed to be either fixed object or leaf object. Further, properties of the at least one object are defined.

The device description framework (DDF) is employed for generating at least a part of a management tree comprising a plurality of objects among which management related information of an mobile communication enabled device is distributed.

(Fig. 5a)

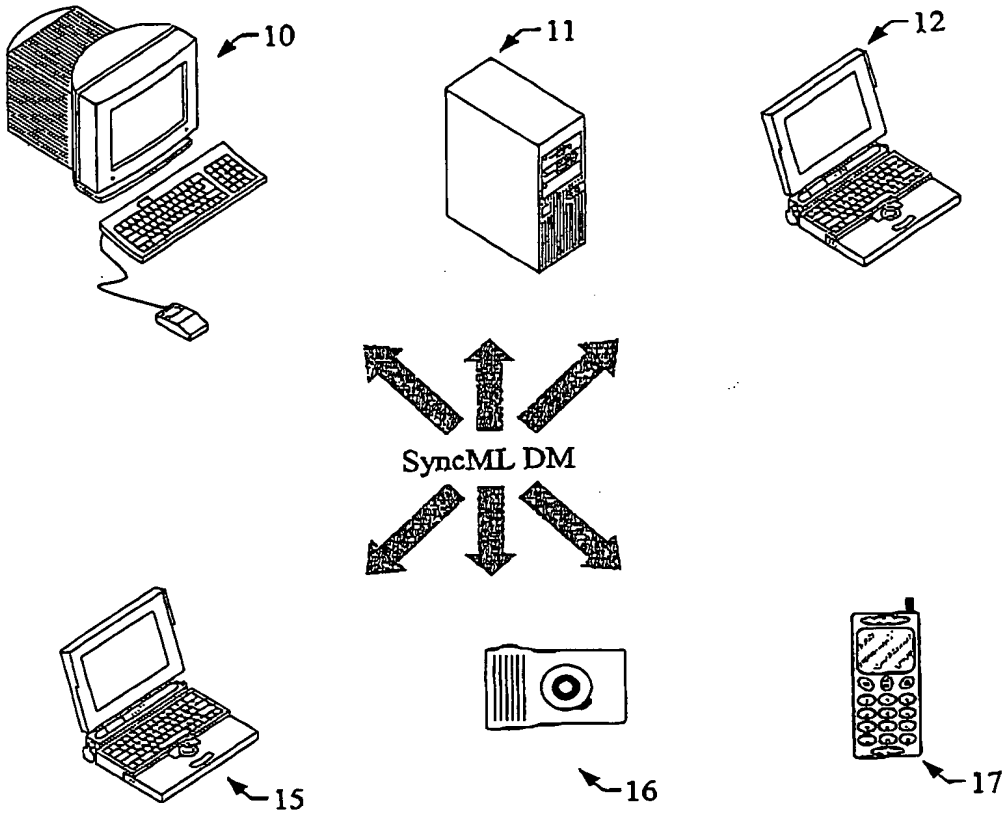


Fig. 1a

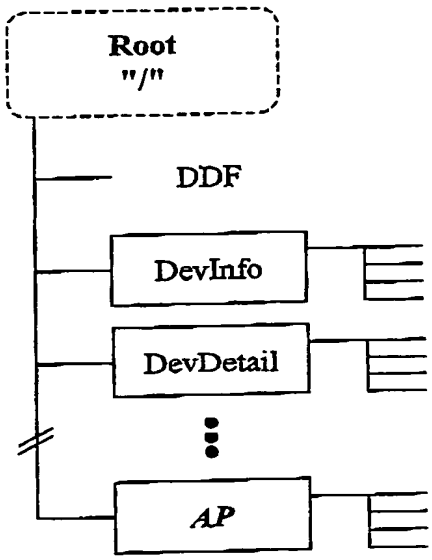


Fig. 1b

Illustrative Representation	Meaning
Fixed	**Fixed (Parent) object:** Permanent object or dynamic object
Run-Time	**Run-Time (Parent) Object:** Dynamic object
Leaf	**Leaf Object:** management object without any child objects
Link	**Link Object:** pointing to another object

Fig. 2a

Character	Meaning
	exactly one occurrences (none character)
+	one or may occurrences
*	zero or more occurrences
?	zero or one occurrences

Fig. 2b

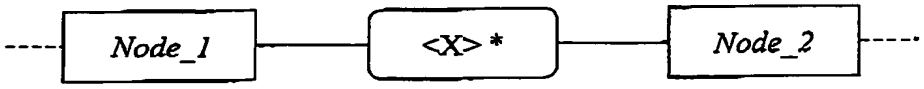


Fig. 3a

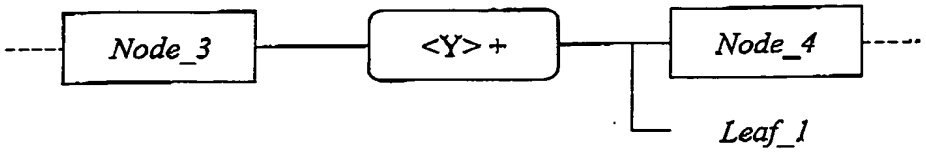


Fig. 3b

Fig. 3c

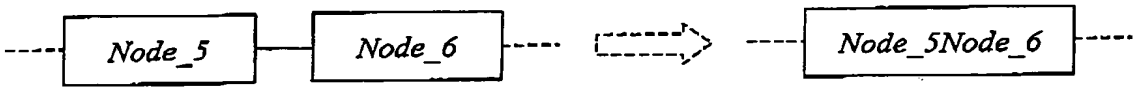


Fig. 3d

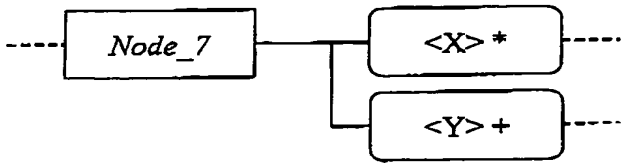


Fig. 3e

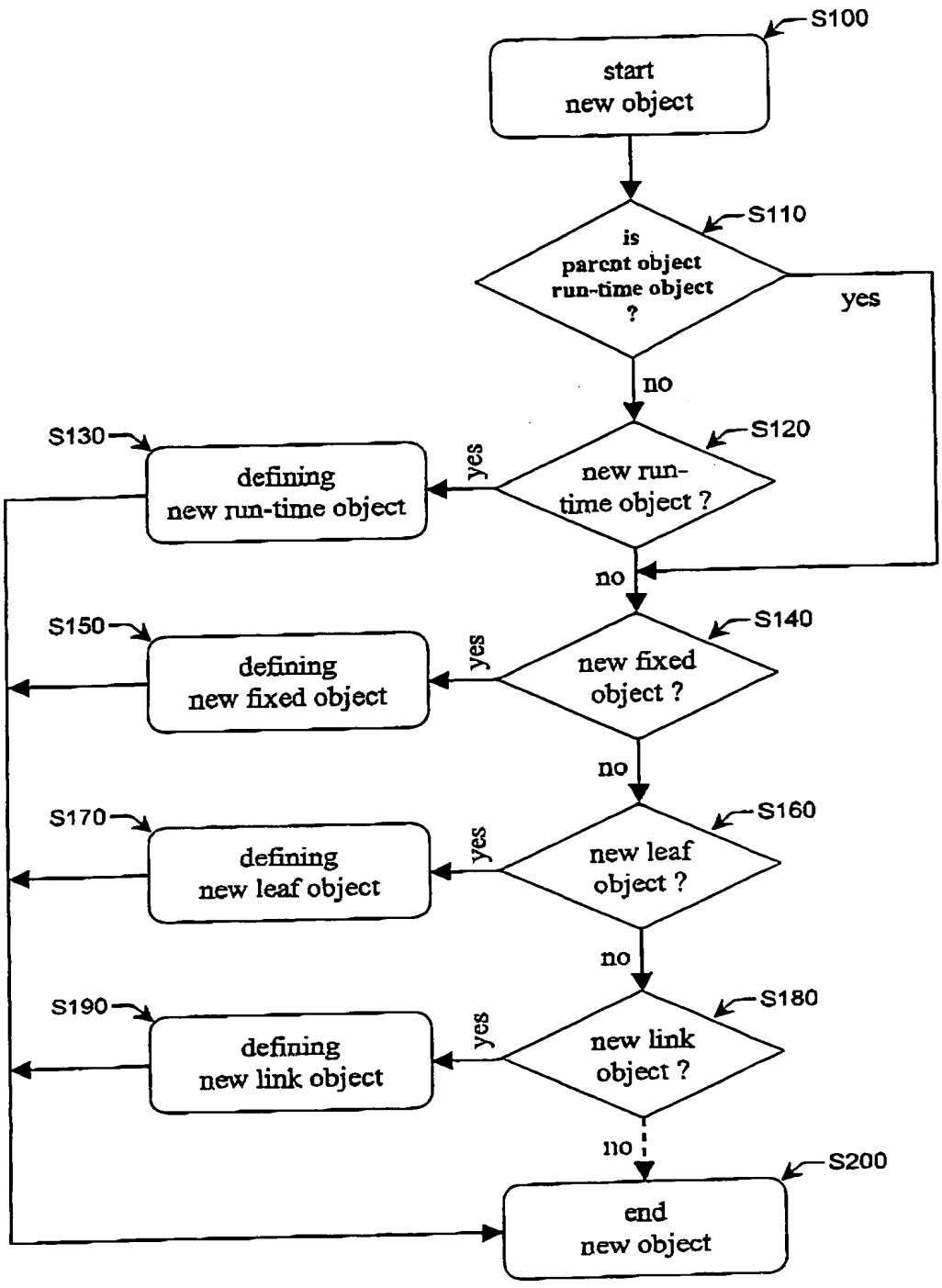

S100
start new object
S110
is parent object run-time object ?
yes
no
S120
new run-time object ?
yes
S130
defining new run-time object
no
S140
new fixed object ?
yes
S150
defining new fixed object
no
S160
new leaf object ?
yes
S170
defining new leaf object
no
S180
new link object ?
yes
S190
defining new link object
no
S200
end new object

Fig. 4a

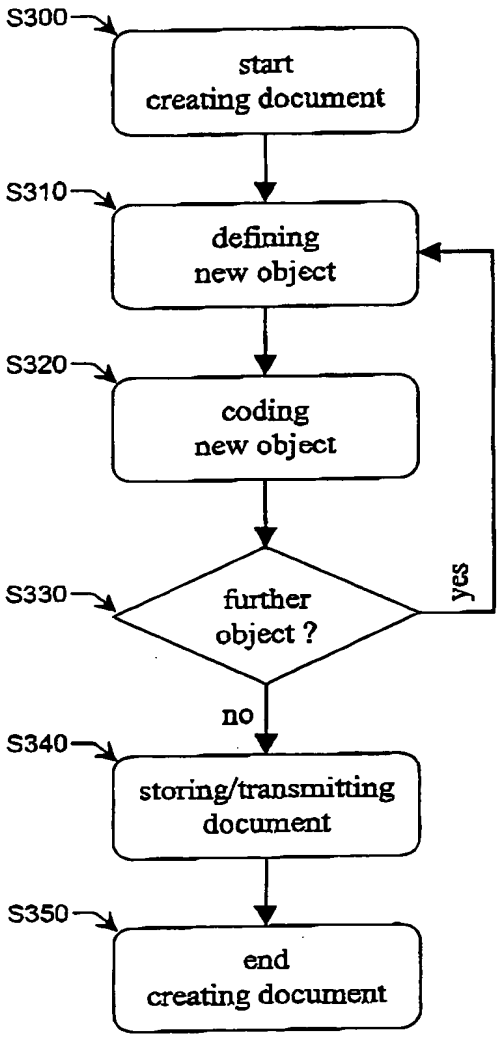

S300
start creating document
S310
defining new object
S320
coding new object
S330
further object ?
yes
no
S340
storing/transmitting document
S350
end creating document

Fig. 4b

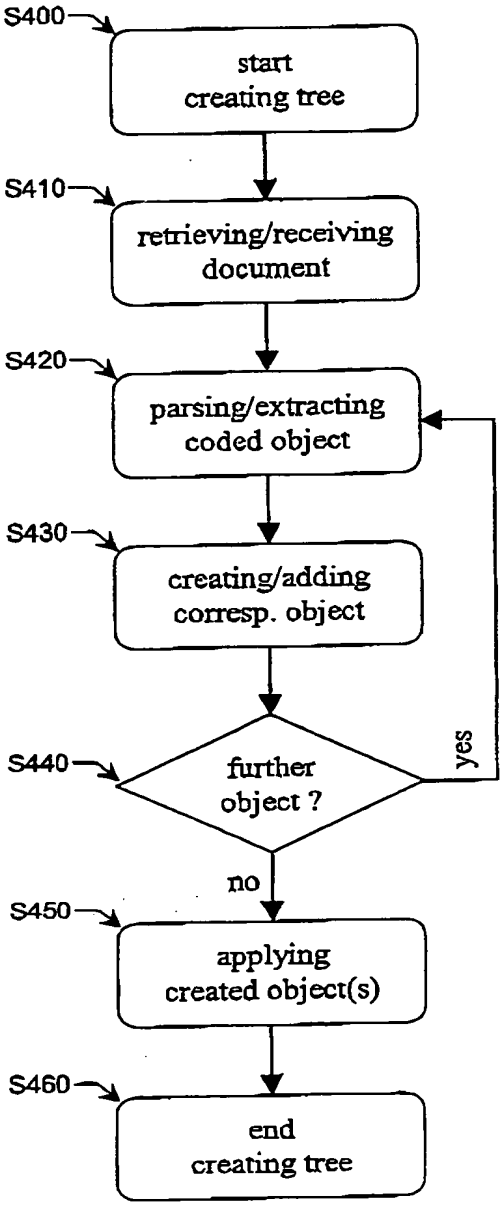

S400
start creating tree
S410
retrieving/receiving document
S420
parsing/extracting coded object
S430
creating/adding corresp. object
S440
further object ?
yes
no
S450
applying created object(s)
S460
end creating tree

Fig. 4c

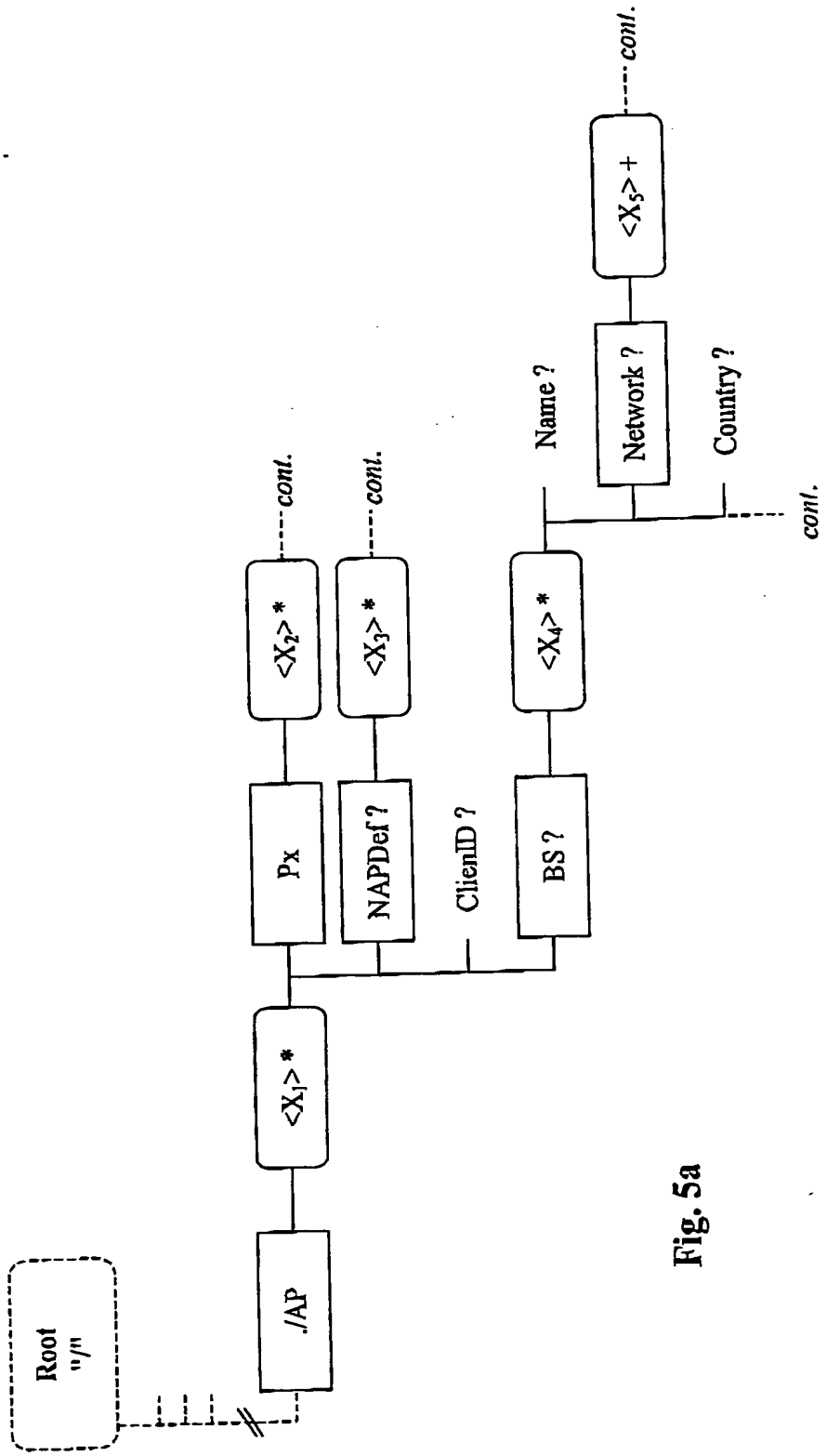

Root "/"
./AP
<X1>*
Px
<X2>*
cont.
NAPDef ?
<X3>*
cont.
ClienID ?
BS ?
<X4>*
Name ?
Network ?
<X5>+
cont.
Country ?
cont.

Fig. 5a

```
001 : <?xml version="1.0" encoding="UTF-8"?>
002 : <!DOCTYPE MgmtTree SYSTEM "...syncml_dm_ddf.dtd">
003 : <MgmtTree>
004 :    <VerDTD>1.1</VerDTD>
005 :    <Man>--Nokia--</Man>
006 :    <Mod>--The device model--</Mod>
007 :    <Node>
008 :       <NodeName>AP</NodeName>
009 :       <Path>./</Path>
010 :       <DFProperties>
011 :          <AccessType> <Get/> </AccessType>
012 :          <Description>Access Point settings</Description>
013 :          <DFFormat> <node/> </DFFormat>
014 :          <Occurrence> <One/> </Occurrence>
015 :          <Scope> <Permanent/> </Scope>
016 :          <DFTitle>Access Point object</DFTitle>
017 :       </DFProperties>
018 :       <Node>
019 :          <NodeName/>
020 :          <DFProperties>
021 :             <AccessType> <Add/> <Delete/> <Get/> <Replace/> </AccessType>
022 :             <DFFormat> <node/> </DFFormat>
023 :             <Occurrence> <ZeroOrMore/> </Occurrence>
024 :             <Scope> <Dynamic/> </Scope>
025 :             <DFTitle>"Name" of node X</DFTitle>
026 :          </DFProperties>
027 :          <Node>
028 :             <NodeName>Px</NodeName>
029 :             <DFProperties>
030 :                <AccessType> <Get/> </AccessType>
031 :                <DFFormat> <node/> </DFFormat>
032 :                <Occurrence> <One/> </Occurrence>
033 :                <Scope> <Dynamic/> </Scope>
034 :                <DFTitle>All Proxy objects</DFTitle>
035 :             </DFProperties>
036 :             ...
037 :          </Node>
038 :          <Node>
039 :             <NodeName>NAPDef</NodeName>
040 :             <DFProperties>
041 :                <AccessType> <Get/> </AccessType>
042 :                <DFFormat> <node/> </DFFormat>
043 :                <Occurrence> <ZeroOrOne/> </Occurrence>
044 :                <Scope> <Dynamic/> </Scope>
045 :                <DFTitle>All NAP Definition objects</DFTitle>
046 :             </DFProperties>
047 :             ...
048 :          </Node>
049 :          <Node>
050 :             <NodeName>ClientID</NodeName>
051 :             <DFProperties>
052 :                <AccessType> <Add/> <Delete/> <Get/> <Replace/> </AccessType>
053 :                <DFFormat> <chr/> </DFFormat>
```

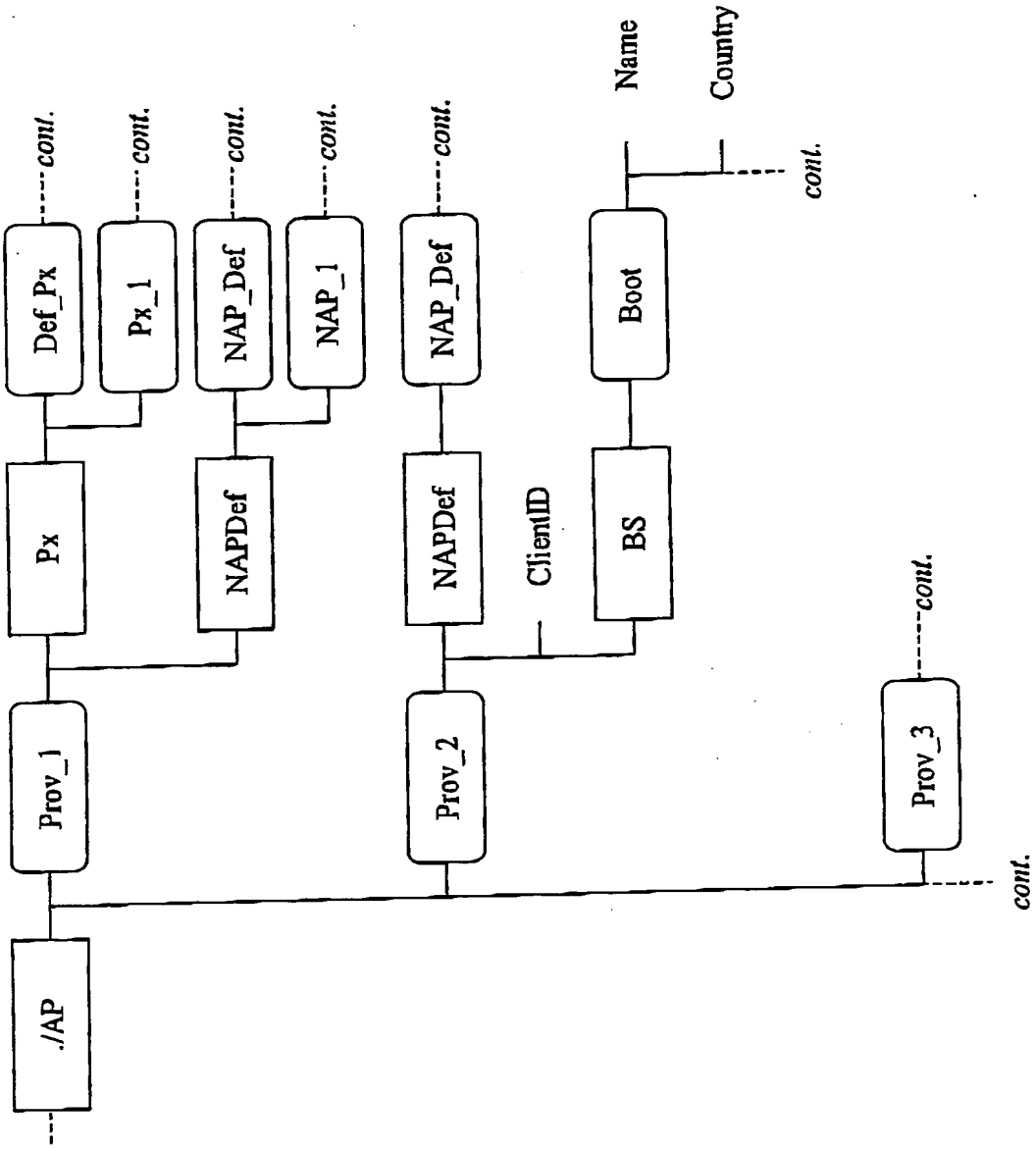
```
001 : <?xml version="1.0" encoding="UTF-8"?>
002 : <!DOCTYPE MgmtTree SYSTEM "... syncml_dm_ddf.dtd">
003 : <MgmtTree>
004 :   <VerDTD>1.1</VerDTD>
005 :   <Man>--Nokia--</Man>
006 :   <Mod>--The device model--</Mod>
007 :   <Node>
008 :     <NodeName>AP</NodeName>
009 :     <Path>./</Path>
010 :     <DFProperties>
011 :       <AccessType> <Get/> </AccessType>
012 :       <Description>Access Point settings</Description>
013 :       <DFFormat> <node/> </DFFormat>
014 :       <Occurrence> <One/> </Occurrence>
015 :       <Scope> <Permanent/> </Scope>
016 :       <DFTitle>Access Point object</DFTitle>
017 :     </DFProperties>
018 :     <Node>
019 :       <NodeName/>
020 :       <DFProperties>
021 :         <AccessType> <Add/> <Delete/> <Get/> <Replace/> </AccessType>
022 :         <DFFormat> <node/> </DFFormat>
023 :         <Occurrence> <ZeroOrMore/> </Occurrence>
024 :         <Scope> <Dynamic/> </Scope>
025 :         <DFTitle>"Name" of node X1</DFTitle>
026 :       </DFProperties>
027 :       <Node>
028 :         <NodeName>Px</NodeName>
029 :         <DFProperties>
030 :           <AccessType> <Get/> </AccessType>
031 :           <DFFormat> <node/> </DFFormat>
032 :           <Occurrence> <One/> </Occurrence>
033 :           <Scope> <Dynamic/> </Scope>
034 :           <DFTitle>All Proxy objects</DFTitle>
035 :         </DFProperties>
036 :         ...
037 :       </Node>
038 :       <Node>
039 :         <NodeName>NAPDef</NodeName>
040 :         <DFProperties>
041 :           <AccessType> <Get/> </AccessType>
042 :           <DFFormat> <node/> </DFFormat>
043 :           <Occurrence> <ZeroOrOne/> </Occurrence>
044 :           <Scope> <Dynamic/> </Scope>
045 :           <DFTitle>All NAP Definition objects</DFTitle>
046 :         </DFProperties>
047 :         ...
048 :       </Node>
049 :       <Node>
050 :         <NodeName>ClientID</NodeName>
051 :         <DFProperties>
052 :           <AccessType> <Add/> <Delete/> <Get/> <Replace/> </AccessType>
053 :           <DFFormat> <chr/> </DFFormat>
```

Fig. 5b

```
060 :          <Node>
061 :            <NodeName>BS</NodeName>
062 :            <DFProperties>
063 :               <AccessType> <Get/> </AccessType>
064 :               <DFFormat> <node/> </DFFormat>
065 :               <Occurrence> <ZeroOrOne/> </Occurrence>
066 :               <Scope> <Dynamic/> </Scope>
067 :               <DFTitle>All BootStrap objects</DFTitle>
068 :            </DFProperties>
069 :            <Node>
070 :               <NodeName/>
071 :               <DFProperties>
072 :                  <AccessType> <Add/> <Delete/> <Get/> <Replace/> </AccessType>
073 :                  <DFFormat> <node/> </DFFormat>
074 :                  <Occurrence> <ZeroOrMore/> </Occurrence>
075 :                  <Scope> <Dynamic/> </Scope>
076 :                  <DFTitle>"Name" of node Xs</DFTitle>
077 :               </DFProperties>
078 :               <Node>
079 :                  <NodeName>Name</NodeName>
080 :                  <DFProperties>
081 :                     <AccessType> <Add/> <Delete/> <Get/> <Replace/> </AccessType>
082 :                     <DFFormat> <chr/> </DFFormat>
083 :                     <Occurrence> <ZeroOrOne/> </Occurrence>
084 :                     <Scope> <Dynamic/> </Scope>
085 :                     <DFTitle>Displayable name</DFTitle>
086 :                     <DFType> <MIME>text/plain</MIME> </DFType>
087 :                  </DFProperties>
088 :               </Node>
089 :               <Node>
090 :                  <NodeName>Network</NodeName>
091 :                  <DFProperties>
092 :                     <AccessType> <Get/> </AccessType>
093 :                     <DFFormat> <node/> </DFFormat>
094 :                     <Occurrence> <ZeroOrOne/> </Occurrence>
095 :                     <Scope> <Dynamic/> </Scope>
096 :                     <DFTitle>All Network objects</DFTitle>
097 :                  </DFProperties>
098 :                  ...
099 :               </Node>
100 :               <Node>
101 :                  <NodeName>Country</NodeName>
102 :                  <DFProperties>
103 :                     <AccessType> <Add/> <Delete/> <Get/> <Replace/> </AccessType>
104 :                     <DFFormat> <chr/> </DFFormat>
105 :                     <Occurrence> <ZeroOrOne/> </Occurrence>
106 :                     <Scope> <Dynamic/> </Scope>
107 :                     <DFTitle>Country code for BS objects</DFTitle>
108 :                     <DFType> <MIME>text/plain</MIME> </DFType>
109 :                  </DFProperties>
110 :               </Node>
111 :               ...
112 :            </Node>
113 :          </Node>
```

Fig. 5c

Fig. 5d


/AP
Prov_1
Prov_2
Prov_3
Px
NAPDef
NAPDef
BS
Boot
Def_Px
Px_1
NAP_Def
NAP_1
NAP_Def
Name
Country
ClientID
cont.
cont.
cont.
cont.
cont.
cont.
cont.
cont.

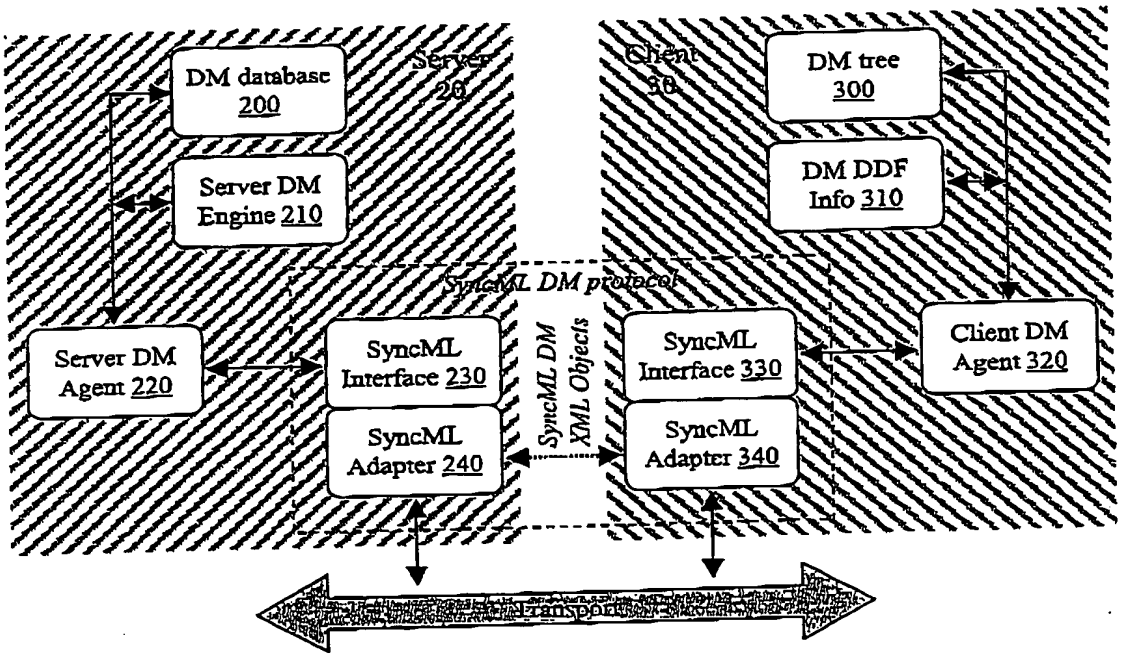

Server 20
DM database 200
Server DM Engine 210
Server DM Agent 220
SyncML Interface 230
SyncML Adapter 240
Client 30
DM tree 300
DM DDF Info 310
SyncML DM protocol
SyncML DM XML Objects
SyncML Interface 330
SyncML Adapter 340
Client DM Agent 320
Transport

Fig. 6